PHP Manuál

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Zeev Suraski

Andrei Zmievski

Jouni Ahto

Vydáno

Stig Sæther Bakken

Egon Schmid

PHP Manuál

Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Zeev Suraski, Andrei Zmievski, a Jouni Ahto

Vydáno Stig Sæther Bakken

Vydáno Egon Schmid

Vydáno 15-04-2001

Copyright

Obsah

Úvod

PHP, což znamená "PHP: Hypertext Preprocessor", je skriptovací jazyk vkládaný do HTML. Velká část jeho syntaxe je vypůjčená z C, Javy a Perlu. Několik vlastností je specifických pro PHP. Cílem tohoto jazyka je je umožnit webovým vývojářum rychle psát dynamicky generované stránky.

About this Manual

Tento manuál je napsán v XML s použitím DocBook XML DTD (http://www.nwalsh.com/docbook/xml/), s DSSSL (http://www.jclark.com/dsssl/) (Document Style and Semantics Specification Language) použitým na formátováni. Nástroje použité na formátování HTML, TeX a RTF verzí jsou Jade (http://www.jclark.com/jade/), napsaný Jamesem Clarkem (http://www.jclark.com/bio.htm) a The Modular DocBook Stylesheets (http://nwalsh.com/docbook/dsssl/) napsané Normanem Walshem (http://nwalsh.com/). Systém dokumentace PHP udržuje Stig Sæther Bakken (mailto:stig@php.net).

Tento manuál si můžete stáhnout v různých jazycích a formátech, včetně PDF, prostého textu, HTML, WinHelpu a RTF z http://www.php.net/docs.php.

Denní snímky HTML verzí manuálu, včetně překladů, jsou k dispozici na http://snaps.php.net/manual/.

Další informace o downloadu XML zdrojového kódu této dokumentace najdete na http://cvs.php.net/. Dokumentace je uložena v `phpdoc` modulu.

Část I. Začínáme

Kapitola 1. Introduction

What is PHP?

PHP (officially "PHP: Hypertext Preprocessor") is a server-side HTML-embedded scripting language.

Simple answer, but what does that mean? An example:

Příklad 1-1. An introductory example

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
        echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Notice how this is different from a CGI script written in other languages like Perl or C – instead of writing a program with lots of commands to output HTML, you write an HTML script with a some embedded code to do something (in this case, output some text). The PHP code is enclosed in special start and end tags that allow you to jump into and out of PHP mode.

What distinguishes PHP from something like client-side Javascript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

What can PHP do?

At the most basic level, PHP can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies.

Perhaps the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. The following databases are currently supported:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

PHP also has support for talking to other services using protocols such as IMAP, SNMP, NNTP, POP3, HTTP and countless others. You can also open raw network sockets and interact using other protocols.

A brief history of PHP

PHP was conceived sometime in the fall of 1994 by Rasmus Lerdorf (mailto:rasmus@php.net). Early non-released versions were used on his home page to keep track of who was looking at his online resume. The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools. It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff. The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

It is difficult to give any hard statistics, but it is estimated that by late 1996 PHP/FI was in use on at least 15,000 web sites around the world. By mid-1997 this number had grown to over 50,000. Mid-1997 also saw a change in the development of PHP. It changed from being Rasmus' own pet project that a handful of people had contributed to, to being a much more organized team effort. The parser was rewritten from scratch by Zeev Suraski and Andi Gutmans and this new parser formed the basis for PHP Version 3. A lot of the utility code from PHP/FI was ported over to PHP 3 and a lot of it was completely rewritten.

The latest version (PHP 4) uses the Zend (http://www.zend.com/) scripting engine to deliver higher performance, supports an even wider array of third-party libraries and extensions, and runs as a native server module with all of the popular web servers.

Today (1/2001) PHP 3 or PHP 4 now ships with a number of commercial products such as Red Hat's Stronghold web server. A conservative estimate based on an extrapolation from numbers provided by Netcraft (http://www.netcraft.com/) (see also Netcraft Web Server Survey (http://www.netcraft.com/survey/)) would be that PHP is in use on over 5,100,000 sites around the world. To put that in perspective, that is slightly more sites than run Microsoft's IIS server on the Internet (5.03 million).

Kapitola 2. Installation

Downloading the latest version

The source code, and binary distributions for some platforms (including Windows), can be found at `http://www.php.net/`. We recommend you to choose mirror (http://www.php.net/mirrors.php) nearest to you for downloading the distributions.

Installation on UNIX systems

This section will guide you through the general configuration and installation of PHP on unix systems. Be sure to investigate any sections specific to your platform or web server before you begin the process.

Prerequisite knowledge and software:

- Basic UNIX skills (being able to operate "make" and a C compiler, if compiling)
- An ANSI C compiler (if compiling)
- flex (for compiling)
- bison (for compiling)
- A web server
- Any module specific components (such as gd, pdf libs, etc.)

There are several ways to install PHP for the Unix platform, either with a compile and configure process, or through various pre-packaged methods. This documentation is mainly focused around the process of compiling and configuring PHP.

The initial PHP setup and configuration process is controlled by the use of the commandline options of the `configure` script. This page outlines the usage of the most common options, but there are many others to play with. Check out the Complete list of configure options for an exhaustive rundown. There are several ways to install PHP:

- As an Apache module
- As an fhttpd module
- For use with AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, or Zeus.
- As a CGI executable

Apache Module Quick Reference

PHP can be compiled in a number of different ways, but one of the most popular is as an Apache module. The following is a quick installation overview.

Příklad 2-1. Quick Installation Instructions for PHP 4 (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure -prefix=/www
7.  cd ../php-x.x.x
8.  ./configure -with-mysql -with-apache=../apache_1.3.x -enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure -activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
```

```
17. Edit your httpd.conf or srm.conf file and add:
      AddType application/x-httpd-php .php

18. Use your normal procedure for restarting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Building

When PHP is configured, you are ready to build the CGI executable. The command **make** should take care of this. If it fails and you can't figure out why, see the Problems section.

Unix/Linux installs

This section contains notes and hints specific to installing PHP on Linux distributions.

Using Packages

Many Linux distributions have some sort of package installation, such as RPM. This can assist in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build php and/or your webserver. If you are unfamiliar with building and compiling your own software, it is worth checking to see whether somebody has already built a packaged version of PHP with the features you need.

Unix/HP-UX installs

This section contains notes and hints specific to installing PHP on HP-UX systems.

Příklad 2-2. Installation Instructions for HP-UX 10

```
From: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
(These tips are for PHP 4.0.4 and Apache v1.3.9)

So you want to install PHP and Apache on a HP-UX 10.20 box?

1. You need gzip, download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z
uncompress the file and install using swinstall

2. You need gcc, download a binary distribution from
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz
gunzip this file and install gcc using swinstall.

3. You need the GNU binutils, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
gunzip and install using swinstall.

4. You now need bison, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
install as above.

5. You now need flex, you need to download the source from one of the
http://www.gnu.org mirrors. It is in the non-gnu directory of the ftp site.
Download the file, gunzip, then tar -xvf it. Go into the newly created flex
directory and do a ./configure, then a make, and then a make install

If you have errors here, it's probably because gcc etc. are not in your
PATH so add them to your PATH.
```

```
Right, now into the hard stuff.

6. Download the PHP and apache sources.

7. gunzip and tar -xvf them.

We need to hack a couple of files so that they can compile ok.

8. Firstly the configure file needs to be hacked because it seems to lose
track of the fact that you are a hpux machine, there will be a
better way of doing this but a cheap and cheerful hack is to put
    lt_target=hpux10.20
on line 47286 of the configure script.

9. Next, the Apache GuessOS file needs to be hacked. Under
apache_1.3.9/src/helpers change line 89 from
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
to:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"

10. You cannot install PHP as a shared object under HP-UX so you must compile
it as a static, just follow the instructions at the Apache page.

11. PHP and apache should have compiled OK, but Apache won't start. you need
to create a new user for Apache, eg www, or apache. You then change lines 252
and 253 of the conf/httpd.conf in Apache so that instead of
    User nobody
    Group nogroup
you have something like
    User www
    Group sys

This is because you can't run Apache as nobody under hp-ux.
Apache and PHP should then work.

Hope this helps somebody,
Paul Mckay.
```

Unix/Solaris installs

This section contains notes and hints specific to installing PHP on Solaris systems.

Required software

Solaris installs often lack C compilers and their related tools. The required software is as follows:

- gcc (recommended, other C compilers may work)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar

In addition, you will need to install (and possibly compile) any additional software specific to your configuration (such as Oracle or MySQL.

Using Packages

You can simplify the Solaris install process by using pkgadd to install most of your needed components.

Unix/OpenBSD installs

This section contains notes and hints specific to installing PHP on OpenBSD (http://www.openbsd.org/).

Using Ports

This is the recommended method of installing PHP on OpenBSD, as it will have the latest patches and security fixes applied to it by the maintainers. To use this method, ensure that you have a recent ports tree (http://www.openbsd.org/ports.html). Then simply find out which flavors you wish to install, and issue the **make install** command. Below is an example of how to do this.

Příklad 2-3. OpenBSD Ports Install Example

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (choose which flavors you want from the list)
$ FLAVOR="imap gettext ldap mysql gd pdflib" make install
$ php4-enable
```

Using Packages

There are pre-compiled packages available for your release of OpenBSD (http://www.openbsd.org/). These integrate automatically with the version of Apache installed with the OS. However, since there are a large number of options (called *flavors*) available for this port, you may find it easier to compile it from source using the ports tree. Read the packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages) manual page for more information in what packages are available.

Unix/Mac OS X installs

This section contains notes and hints specific to installing PHP on Mac OS X.

Using Packages

There are a few pre-packaged and pre-compiled versions of PHP for Mac OS X. This can help in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your web server yourself. If you are unfamiliar with building and compiling your own software, it's worth checking whether somebody has already built a packaged version of PHP with the features you need. Lightyear Design (http://homepage.mac.com/LightyearDesign/MacOSX/Packages/) offers a pre-built version of PHP for OS X, as does Tenon Intersystems (http://www.tenon.com/products/webten/).

Compiling for OS X server

There are two slightly different versions of Mac OS X, client and server. The following is for OS X Server.

Příklad 2-4. Mac OS X server install

```
1. Get the latest distributions of Apache and PHP
2. Untar them, and run the configure program on Apache like so.
    ./configure -exec-prefix=/usr \
```

```
    -localstatedir=/var \
    -mandir=/usr/share/man \
    -libexecdir=/System/Library/Apache/Modules \
    -iconsdir=/System/Library/Apache/Icons \
    -includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    -enable-shared=max \
    -enable-module=most \
    -target=apache

4. You may also want to add this line:
    setenv OPTIM=-O2
    If you want the compiler to do some optimization.

5. Next, go to the PHP 4 source directory and configure it.
    ./configure -prefix=/usr \
    -sysconfdir=/etc \
    -localstatedir=/var \
    -mandir=/usr/share/man \
    -with-xml \
    -with-apache=/src/apache_1.3.12

    If you have any other addiitons (MySQL, GD, etc.), be sure to add
    them here. For the -with-apache string, put in the path to your
    apache source directory, for example "/src/apache_1.3.12".
6. make
7. make install
    This will add a directory to your Apache source directory under
    src/modules/php4.

8. Now, reconfigure Apache to build in PHP 4.
    ./configure -exec-prefix=/usr \
    -localstatedir=/var \
    -mandir=/usr/share/man \
    -libexecdir=/System/Library/Apache/Modules \
    -iconsdir=/System/Library/Apache/Icons \
    -includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    -enable-shared=max \
    -enable-module=most \
    -target=apache \
    -activate-module=src/modules/php4/libphp4.a

    You may get a message telling you that libmodphp4.a is out of date.
    If so, go to the src/modules/php4 directory inside your apache
    source directory and run this command:

    ranlib libmodphp4.a

    Then go back to the root of the apache source directory and run the
    above configure command again. That'll bring the link table up to
    date.

9. make

10. make install

11. copy and rename the php.ini-dist file to your "bin" directory from your
    PHP 4 source directory:
    cp php.ini-dist /usr/local/bin/php.ini

    or (if your don't have a local directory)

    cp php.ini-dist /usr/bin/php.ini
```

Other examples for Mac OS X client (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) and Mac OS X server (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) are available at Stepwise (http://www.stepwise.com/).

Complete list of configure options

> **Poznámka:** These are only used at compile time. If you want to alter PHP's runtime configuration, please see the chapter on Configuration.

The following is a complete list of options supported by the PHP 3 and PHP 4 `configure` scripts, used when compiling in Unix-like environments. Some are available in PHP 3, some in PHP 4, and some in both, as noted. There are many options the names of which have changed between PHP 3 and PHP 4, but which accomplish the same things. These entries are cross-referenced to each other, so if you have a problem getting your PHP 3-era configuration options to work, check here to see whether the names have changed.

- Database
- Ecommerce
- Graphics
- Miscellaneous
- Networking
- PHP Behaviour
- Server
- Text and language
- XML

Database

`-with-adabas[=DIR]`

PHP 3, PHP 4: Include Adabas D support. DIR is the Adabas base install directory, defaults to /usr/local.

Adabas home page (http://www.adabas.com/)

`-enable-dba=shared`

PHP 3: Option not available in PHP 3

PHP 4: Build DBA as a shared module

`-enable-dbase`

PHP 3: Option not available; use –with-dbase instead.

PHP 4: Enable the bundled dbase library. No external libraries are required.

`-with-dbase`

PHP 3: Include the bundled dbase library. No external libraries are required.

PHP 4: Option not available; use –enable-dbase instead.

`-with-db2[=DIR]`

PHP 3, PHP 4: Include Berkeley DB2 support

`-with-db3[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include Berkeley DB3 support

-with-dbm[=DIR]

PHP 3, PHP 4: Include DBM support

-with-dbmaker[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include DBMaker support. DIR is the DBMaker base install directory, defaults to where the latest version of DBMaker is installed (such as /home/dbmaker/3.6).

-with-empress[=DIR]

PHP 3, PHP 4: Include Empress support. DIR is the Empress base install directory, defaults to $EMPRESSPATH

-enable-filepro

PHP 3: Option not available; use –with-filepro instead.

PHP 4: Enable the bundled read-only filePro support. No external libraries are required.

-with-filepro

PHP 3: Include the bundled read-only filePro support. No external libraries are required.

PHP 4: Option not available; use –enable-filepro instead.

-with-gdbm[=DIR]

PHP 3, PHP 4: Include GDBM support

-with-hyperwave

PHP 3, PHP 4: Include Hyperwave support

-with-ibm-db2[=DIR]

PHP 3, PHP 4: Include IBM DB2 support. DIR is the DB2 base install directory, defaults to `/home/db2inst1/sqllib`.

IBM DB2 home page (http://www.ibm.com/db2/)

-with-informix[=DIR]

PHP 3, PHP 4: Include Informix support. DIR is the Informix base install directory, defaults to nothing.

-with-ingres[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Ingres II support. DIR is the Ingres base directory (default /II/ingres)

-with-interbase[=DIR]

PHP 3, PHP 4: Include InterBase support. DIR is the InterBase base install directory, which defaults to `/usr/interbase`.

Interbase functions

Interbase home page (http://www.interbase.com/)

-with-ldap[=DIR]

PHP 3: Include LDAP support. DIR is the LDAP base install directory. Defaults to `/usr` and `/usr/local`

PHP 4: Include LDAP support. DIR is the LDAP base install directory.

This provides LDAP (Lightweight Directory Access Protocol support). The parameter is the LDAP base install directory, defaults to `/usr/local/ldap`.

More information about LDAP can be found in RFC1777 (http://www.faqs.org/rfcs/rfc1777.html) and RFC1778 (http://www.faqs.org/rfcs/rfc1778.html).

-with-msql[=DIR]

PHP 3, PHP 4: Enables mSQL support. The parameter to this option is the mSQL install directory and defaults to `/usr/local/Hughes`. This is the default directory of the mSQL 2.0 distribution. **configure** automatically detects which mSQL version you are running and PHP supports both 1.0 and 2.0, but if you compile PHP with mSQL 1.0, you can only access mSQL 1.0 databases, and vice-versa.

See also mSQL Configuration Directives in the configuration file.

mSQL home page (http://www.hughes.com.au/)

-with-mysql[=DIR]

PHP 3: Include MySQL support. DIR is the MySQL base install directory, defaults to searching through a number of common places for the MySQL files.

PHP 4: Include MySQL support. DIR is the MySQL base directory. If unspecified, the bundled MySQL library will be used. This option is turned on by default.

See also MySQL Configuration Directives in the configuration file.

MySQL home page (http://www.mysql.com/)

-with-ndbm[=DIR]

PHP 3, PHP 4: Include NDBM support

-with-ovrimos

PHP 3, PHP 4: Include Ovrimos support.

-with-oci8[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Oracle-oci8 support. Default DIR is ORACLE_HOME.

-with-oracle[=DIR]

PHP 3: Include Oracle database support. DIR is Oracle's home directory, defaults to $ORACLE_HOME.

PHP 4: Include Oracle-oci7 support. Default DIR is ORACLE_HOME.

Includes Oracle support. Has been tested and should be working at least with Oracle versions 7.0 through 7.3. The parameter is the ORACLE_HOME directory. You do not have to specify this parameter if your Oracle environment has been set up.

Oracle home page (http://www.oracle.com/)

-with-pgsql[=DIR]

PHP 3: Include PostgresSQL support. DIR is the PostgresSQL base install directory, which defaults to `/usr/local/pgsql`.

PHP 4: Include PostgreSQL support. DIR is the PostgreSQL base install directory, which defaults to `/usr/local/pgsql`. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

See also Postgres Configuration Directives in the configuration file.

PostgreSQL home page (http://www.postgresql.org/)

-with-solid[=DIR]

PHP 3, PHP 4: Include Solid support. DIR is the Solid base install directory, defaults to /usr/local/solid

Solid home page (http://www.solidtech.com/)

-with-sybase-ct[=DIR]

PHP 3, PHP 4: Include Sybase-CT support. DIR is the Sybase home directory, defaults to /home/sybase.

See also Sybase-CT Configuration Directives in the configuration file.

-with-sybase[=DIR]

PHP 3, PHP 4: Include Sybase-DB support. DIR is the Sybase home directory, which defaults to `/home/sybase`.

See also Sybase Configuration Directives in the configuration file.

Sybase home page (http://www.sybase.com/)

-with-openlink[=DIR]

PHP 3, PHP 4: Include OpenLink ODBC support. DIR is the OpenLink base install directory, defaults to /usr/local/openlink.

OpenLink Software's home page (http://www.openlinksw.com/)

-with-iodbc[=DIR]

PHP 3, PHP 4: Include iODBC support. DIR is the iODBC base install directory, defaults to `/usr/local`.

This feature was first developed for iODBC Driver Manager, a freely redistributable ODBC driver manager which runs under many flavors of UNIX.

FreeODBC home page (http://users.ids.net/~bjepson/freeODBC/) or iODBC home page (http://www.iodbc.org/)

-with-custom-odbc[=DIR]

PHP 3, PHP 4: Includes support for an arbitrary custom ODBC library. The parameter is the base directory and defaults to `/usr/local`.

This option implies that you have defined CUSTOM_ODBC_LIBS when you run the configure script. You also must have a valid odbc.h header somewhere in your include path. If you don't have one, create it and include your specific header from there. Your header may also require some extra definitions, particularly when it is multiplatform. Define them in CFLAGS.

For example, you can use Sybase SQL Anywhere on QNX as following: `CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure -with-custom-odbc=/usr/lib/sqlany50`

-disable-unified-odbc

PHP 3: Disable unified ODBC support. Only applicable if iODBC, Adabas, Solid, Velocis or a custom ODBC interface is enabled.

PHP 4: Option not available in PHP 4

The Unified ODBC module, which is a common interface to all the databases with ODBC-based interfaces, such as Solid, IBM DB2 and Adabas D. It also works for normal ODBC libraries. Has been tested with iODBC, Solid, Adabas D, IBM DB2 and Sybase SQL Anywhere. Requires that one (and only one) of these extensions or the Velocis extension is enabled, or a custom ODBC library specified. This option is only applicable if one of the following options is used: –with-iodbc, –with-solid, –with-ibm-db2, –with-adabas, –with-velocis, or –with-custom-odbc.

See also Unified ODBC Configuration Directives in the configuration file.

-with-unixODBC[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include unixODBC support. DIR is the unixODBC base install directory, defaults to /usr/local.

-with-velocis[=DIR]

PHP 3, PHP 4: Include Velocis support. DIR is the Velocis base install directory, defaults to /usr/local/velocis.

Velocis home page (http://www.raima.com/)

Ecommerce

-with-ccvs[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Compile CCVS support into PHP 4. Please specify your CCVS base install directory as DIR.

-with-mck[=DIR]

PHP 3: Include Cybercash MCK support. DIR is the cybercash mck build directory, which defaults to `/usr/src/mck-3.2.0.3-linux`. For help, look in `extra/cyberlib`.

PHP 4: Option not available; use –with-cybercash instead.

-with-cybercash[=DIR]

PHP 3: Option not available; use –with-mck instead.

PHP 4: Include CyberCash support. DIR is the CyberCash MCK install directory.

-with-pfpro[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Verisign Payflow Pro support

Graphics

-enable-freetype-4bit-antialias-hack

PHP 3: Option not available in PHP 3

PHP 4: Include support for FreeType2 (experimental).

-with-gd[=DIR]

PHP 3: Include GD support (DIR is GD's install dir).

PHP 4: Include GD support (DIR is GD's install dir). Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

-without-gd

PHP 3, PHP 4: Disable GD support.

-with-imagick[=DIR]

PHP 3: Include ImageMagick support. DIR is the install directory, and if left out, PHP will try to find it on its own. [experimental]

PHP 4: Option not available in PHP 4

-with-jpeg-dir[=DIR]

PHP 3: jpeg dir for pdflib 2.0

PHP 4: jpeg dir for pdflib 3.x

-with-png-dir[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: png dir for pdflib 3.x

-enable-t1lib

PHP 3: Enable t1lib support.

PHP 4: Option not available; use –with-t1lib instead.

-with-t1lib[=DIR]

PHP 3: Option not available; use –enable-t1lib instead.

PHP 4: Include T1lib support.

-with-tiff-dir[=DIR]

PHP 3: tiff dir for pdflib 2.0

PHP 4: tiff dir for pdflib 3.x

-with-ttf[=DIR]

PHP 3, PHP 4: Include FreeType support

-with-xpm-dir[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: xpm dir for gd-1.8+

Miscellaneous

These are being classified over time, where appropriate.

-with-gmp

PHP 3, PHP 4 : Include GMP support.

-disable-bcmath

PHP 3: Compile without BC arbitrary precision math functions. These functions allow you to operate with numbers outside of the ranges allowed by regular integers and floats; see BCMath Arbitrary Precision Mathematics Functions for more information.

PHP 4: Option not available; bcmath is not compiled in by default. Use –enable-bcmath to compile it in.

-disable-display-source

PHP 3: Compile without displaying source support

PHP 4: Option not available in PHP 4

-disable-libtool-lock

PHP 3: Option not available in PHP 3

PHP 4: avoid locking (might break parallel builds)

-disable-pear

PHP 3: Option not available in PHP 3

PHP 4: Do not install PEAR

-disable-pic

PHP 3: Option not available in PHP 3

PHP 4: Disable PIC for shared objects

-disable-posix

PHP 3: Option not available in PHP 3; use –without-posix instead.

PHP 4: Disable POSIX-like functions

-disable-rpath

PHP 3: Option not available in PHP 3

PHP 4: Disable passing additional runtime library search paths

-disable-session

PHP 3: Option not available in PHP 3

PHP 4: Disable session support

-enable-bcmath

PHP 3: Option not available in PHP 3; bcmath is compiled in by default. Use –disable-bcmath to disable it.

PHP 4: Compile with bc style precision math functions. Read README-BCMATH for instructions on how to get this module installed. These functions allow you to operate with numbers outside of the ranges allowed by regular integers and floats; see BCMath Arbitrary Precision Mathematics Functions for more information.

-enable-c9x-inline

PHP 3: Option not available in PHP 3

PHP 4: Enable C9x-inline semantics

-enable-calendar

PHP 3: Option not available in PHP 3

PHP 4: Enable support for calendar conversion

-enable-debug

PHP 3, PHP 4: Compile with debugging symbols.

-enable-debugger

PHP 3: Compile with remote debugging functions

PHP 4: Option not available in PHP 4

-enable-discard-path

PHP 3, PHP 4: If this is enabled, the PHP CGI binary can safely be placed outside of the web tree and people will not be able to circumvent .htaccess security.

-enable-dmalloc

PHP 3, PHP 4: Enable dmalloc

-enable-exif

PHP 3: Option not available in PHP 3

PHP 4: Enable exif support

-enable-experimental-zts

PHP 3: Option not available in PHP 3

PHP 4: This will most likely break your build

-enable-fast-install[=PKGS]

PHP 3: Option not available in PHP 3

PHP 4: optimize for fast installation [default=yes]

-enable-force-cgi-redirect

PHP 3, PHP 4: Enable the security check for internal server redirects. You should use this if you are running the CGI version with Apache.

-enable-inline-optimization

PHP 3: Option not available in PHP 3

PHP 4: If you have much memory and are using gcc, you might try this.

-enable-libgcc

PHP 3: Option not available in PHP 3

PHP 4: Enable explicitly linking against libgcc

-enable-maintainer-mode

PHP 3, PHP 4: enable make rules and dependencies not useful (and sometimes confusing) to the casual installer

-enable-memory-limit

PHP 3, PHP 4: Compile with memory limit support. [default=no]

-enable-safe-mode

PHP 3, PHP 4: Enable safe mode by default.

-enable-satellite

PHP 3: Option not available in PHP 3

PHP 4: Enable CORBA support via Satellite (Requires ORBit)

-enable-shared[=PKGS]

PHP 3: Option not available in PHP 3

PHP 4: build shared libraries [default=yes]

-enable-sigchild

PHP 3, PHP 4: Enable PHP's own SIGCHLD handler.

-enable-static[=PKGS]

PHP 3: Option not available in PHP 3

PHP 4: build static libraries [default=yes]

-enable-sysvsem

PHP 3, PHP 4: Enable System V semaphore support.

-enable-sysvshm

PHP 3, PHP 4: Enable the System V shared memory support

-enable-trans-sid

PHP 3: Option not available in PHP 3

PHP 4: Enable transparent session id propagation

-with-cdb[=DIR]

PHP 3, PHP 4: Include CDB support

-with-config-file-path=PATH

PHP 3: Sets the path in which to look for php3.ini. Defaults to `/usr/local/lib`

PHP 4: Sets the path in which to look for `php.ini`. Defaults to `/usr/local/lib`

-with-cpdflib[=DIR]

PHP 3: Include ClibPDF support. DIR is the ClibPDF install directory, defaults to /usr/local.

PHP 4: Include cpdflib support (requires cpdflib >= 2). DIR is the cpdfllib install directory, defaults to /usr.

-with-esoob[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Easysoft OOB support. DIR is the OOB base install directory, defaults to /usr/local/easysoft/oob/client.

-with-exec-dir[=DIR]

PHP 3, PHP 4: Only allow executables in DIR when in safe mode defaults to /usr/local/php/bin

-with-fdftk[=DIR]

PHP 3, PHP 4: Include fdftk support. DIR is the fdftk install directory, defaults to /usr/local.

-with-gnu-ld

PHP 3: Option not available in PHP 3

PHP 4: assume the C compiler uses GNU ld [default=no]

-with-icap[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include ICAP support.

-with-imap[=DIR]

PHP 3, PHP 4: Include IMAP support. DIR is the IMAP include and c-client.a directory.

-with-imsp[=DIR]

PHP 3: Include IMSP support (DIR is IMSP's include dir and libimsp.a dir).

PHP 4: Option not available in PHP 4

-with-java[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Java support. DIR is the base install directory for the JDK. This extension can only be built as a shared dl.

-with-kerberos[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Kerberos support in IMAP.

-with-mcal[=DIR]

PHP 3, PHP 4: Include MCAL support.

-with-mcrypt[=DIR]

PHP 3, PHP 4: Include mcrypt support. DIR is the mcrypt install directory.

-with-mhash[=DIR]

PHP 3, PHP 4: Include mhash support. DIR is the mhash install directory.

-with-mm[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include mm support for session storage

-with-mod_charset

PHP 3, PHP 4: Enable transfer tables for mod_charset (Rus Apache).

-with-pdflib[=DIR]

PHP 3: Include pdflib support (tested with 0.6 and 2.0). DIR is the pdflib install directory, which defaults to `/usr/local`.

PHP 4: Include pdflib 3.x support. DIR is the pdflib install directory, which defaults to `/usr/local`.

-enable-shared-pdflib

PHP 3, PHP 4: Activate pdflib as shared librairy.

-with-readline[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include readline support. DIR is the readline install directory.

-with-regex=TYPE

PHP 3: Option not available in PHP 3

PHP 4: regex library type: system, apache, php

-with-servlet[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include servlet support. DIR is the base install directory for the JSDK. This SAPI requires that the Java extension be built as a shared dl.

-with-ming

PHP 3: Option not available in PHP 3

PHP 4: Include Flash 4 support, with Ming

-with-swf[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include swf support

-with-system-regex

PHP 3: Do not use the bundled regex library

PHP 4: (deprecated) Use system regex library

-with-tsrm-pth[=pth-config]

PHP 3: Option not available in PHP 3

PHP 4: Use GNU Pth.

-with-tsrm-pthreads

PHP 3: Option not available in PHP 3

PHP 4: Use POSIX threads (default)

-with-x

PHP 3: use the X Window System

PHP 4: Option not available in PHP 4

-with-bz2[=DIR]

PHP 3, PHP 4: Include support bzip2. DIR is the bzip2 install dir.

-with-zlib-dir[=DIR]

PHP 3: zlib dir for pdflib 2.0 or include zlib support

PHP 4: zlib dir for pdflib 3.x or include zlib support

-with-zlib[=DIR]

PHP 3, PHP 4: Include zlib support (requires zlib >= 1.0.9). DIR is the zlib install directory, defaults to /usr.

-without-pcre-regex

PHP 3: Don't include Perl Compatible Regular Expressions support

PHP 4: Do not include Perl Compatible Regular Expressions support. Use –with-pcre-regex=DIR to specify DIR where PCRE's include and library files are located, if not using bundled library.

-without-posix

PHP 3: Don't include POSIX functions

PHP 4: Option not available in PHP 4; use –disable-posix instead.

Networking

-with-curl[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include CURL support

-enable-ftp

PHP 3: Option not available; use –with-ftp instead.

PHP 4: Enable FTP support

-with-ftp

PHP 3: Include FTP support.

PHP 4: Option not available; use –enable-ftp instead

-disable-url-fopen-wrapper

PHP 3, PHP 4: Disable the URL-aware fopen wrapper that allows accessing files via http or ftp.

Varování

This switch is only available for PHP versions up to 4.0.3, newer versions provide an INI parameter called *allow_url_fopen* instead of forcing you to decide upon this feature at compile time.

-with-mod-dav=DIR

PHP 3, PHP 4: Include DAV support through Apache's mod_dav, DIR is mod_dav's installation directory (Apache module version only!)

-with-openssl[=DIR]

PHP 3, PHP 4: Include OpenSSL support in SNMP.

-with-snmp[=DIR]

PHP 3, PHP 4: Include SNMP support. DIR is the SNMP base install directory, defaults to searching through a number of common locations for the snmp install. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

-enable-ucd-snmp-hack

PHP 3, PHP 4: Enable UCD SNMP hack

-enable-sockets

PHP 3: Option not available in PHP 3

PHP 4: Enable sockets support

-with-yaz[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include YAZ support (ANSI/NISO Z39.50). DIR is the YAZ bin install directory

-enable-yp

PHP 3: Option not available; use –with-yp instead.

PHP 4: Include YP support

-with-yp

PHP 3: Include YP support

PHP 4: Option not available; use –enable-yp instead.

-with-mnogosearch

PHP 3, PHP 4: Include mnoGoSearch support.

PHP Behaviour

-enable-magic-quotes

PHP 3, PHP 4: Enable magic quotes by default.

-disable-short-tags

PHP 3, PHP 4: Disable the short-form <? start tag by default.

-enable-track-vars

PHP 3: Enable GET/POST/Cookie track variables by default.

PHP 4: Option not available in PHP 4; as of PHP 4.0.2, track_vars is always on.

Server

-with-aolserver-src=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the source distribution of AOLserver

-with-aolserver=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the installed AOLserver

-with-apache[=DIR]

PHP 3, PHP 4: Build Apache module. DIR is the top-level Apache build directory, defaults to /usr/local/etc/httpd.

-with-apxs[=FILE]

PHP 3, PHP 4: Build shared Apache module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs.

-enable-versioning

PHP 3: Take advantage of versioning and scoping Provided by Solaris 2.x and Linux

PHP 4: Export only required symbols. See INSTALL for more information

-with-caudium[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a Pike module for use with the Caudium webserver. DIR is the Caudium base directory. If no directory is specified $prefix/caudium/server is used. The prefix is controlled by the –prefix option and is /usr/local per default.

-with-fhttpd[=DIR]

PHP 3, PHP 4: Build fhttpd module. DIR is the fhttpd sources directory, defaults to /usr/local/src/fhttpd.

-with-nsapi=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the installed Netscape

-with-phttpd=DIR

PHP 3: Option not available in PHP 3

PHP 4:

-with-pi3web=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a module for use with Pi3Web.

-with-roxen=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a Pike module. DIR is the base Roxen directory, normally /usr/local/roxen/server.

-enable-roxen-zts

PHP 3: Option not available in PHP 3

PHP 4: Build the Roxen module using Zend Thread Safety.

-with-thttpd=SRCDIR

PHP 3: Option not available in PHP 3

PHP 4:

-with-zeus=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as an ISAPI module for use with Zeus.

Text and language

-with-aspell[=DIR]

PHP 3, PHP 4: Include ASPELL support.

-with-gettext[=DIR]

PHP 3, PHP 4: Include GNU gettext support. DIR is the gettext install directory, defaults to /usr/local

-with-pspell[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include PSPELL support.

-with-recode[=DIR]

PHP 3: Include GNU recode support.

PHP 4: Include recode support. DIR is the recode install directory.

-enable-shmop

PHP 3, PHP 4 : Activate shmop support.

XML

-with-dom[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include DOM support (requires libxml >= 2.0). DIR is the libxml install directory, defaults to `/usr`

-enable-sablot-errors-descriptive

PHP 3: Option not available in PHP 3

PHP 4: Enable Descriptive errors

-with-sablot[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Sablotron support

-enable-wddx

PHP 3: Option not available in PHP 3

PHP 4: Enable WDDX support

-disable-xml

PHP 3: Option not available in PHP 3; XML functionality is not built in by default. Use –with-xml to turn it on.

PHP 4: Disable XML support using bundled expat lib

-with-xml

PHP 3: Include XML support

PHP 4: Option not available; XML support is built in by default. Use –disable-xml to turn it off.

Installation on Windows 9x/Me/NT/2000 systems

There are two main ways to install PHP for Windows: either manually or by using the InstallShield installer.

If you have Microsoft Visual Studio, you can also build PHP from the original source code.

Once you have PHP installed on your Windows system, you may also want to load various extensions for added functionality.

Windows InstallShield

The Windows PHP installer available from the downloads page at http://www.php.net/, this installs the CGI version of PHP and, for IIS, PWS, and Xitami, configures the web server as well.

Install your selected HTTP server on your system and make sure that it works.

Run the executable installer and follow the instructions provided by the installation wizard. Two types of installation are supported - standard, which provides sensible defaults for all the settings it can, and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the `php.ini` file and configure the web server to use PHP. For IIS and also PWS on NT Workstation, a list of all the nodes on the server with script map settings is displayed, and you can choose those nodes to which you wish to add the PHP script mappings.

Once the installation has completed the installer will inform you if you need to restart your system, restart the server, or just start using PHP.

General Installation Steps

This install guide will help you manually install and configure PHP on your Windows 9x/Me/NT/2000 webservers. This guide was compiled by Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us). The original version can be found at http://www.umesd.k12.or.us/php/win32install.html.

This guide provides manual installation support for:

- Personal Web Server 3 and 4 or newer
- Internet Information Server 3 and 4 or newer
- Apache 1.3.x
- OmniHTTPd 2.0b1 and up
- Oreilly Website Pro
- Xitami

PHP 4 for Windows comes in two flavours - a CGI executable (php.exe), and several SAPI modules (for exapmle php4isapi.dll). The latter form is new to PHP 4, and provides significantly improved performance and some new functionality. However, please note that the SAPI modules are *NOT* yet considered to be production quality. The reason for this is that the PHP SAPI modules are using the thread-safe version of the PHP code, which is new to PHP 4, and has not yet been tested and pounded enough to be considered completely stable, and there are actually a few known bugs. On the other hand, some people have reported very good results with the SAPI modules, even though we’re not aware of anyone actually running it on a production site. In short - your mileage may vary; If you need absolute stability, trade the performance of the SAPI modules with the stability of the CGI executable.

If you choose one of the SAPI modules and use Windows 95, be sure to download the DCOM update from the Microsoft DCOM pages (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). For the ISAPI module, an ISAPI 4.0 compliant Web server is required (tested on IIS 4.0, PWS 4.0 and IIS 5.0). IIS 3.0 is *NOT* supported; You should download and install the Windows NT 4.0 Option Pack with IIS 4.0 if you want native PHP support.

The following steps should be performed on all installations before the server specific instructions.

- Extract the distribution file to a directory of your choice. "C:\PHP\" is a good start.

- Copy the file, ’php.ini-dist’ to your ’%WINDOWS%’ directory on Windows 95/98 or to your ’%SYSTEMROOT%’ directory under Windows NT or Windows 2000 and rename it to ’php.ini’. Your ’%WINDOWS%’ or ’%SYSTEMROOT%’ directory is typically:

  c:\windows for Windows 95/98
  c:\winnt or c:\winnt40 for NT/2000 servers

- Edit your ’php.ini’ file:
  - You will need to change the ’extension_dir’ setting to point to your php-install-dir, or where you have placed your ’php_*.dll’ files. ex: c:\php
  - If you are using OmniHTTPd, do not follow the next step. Set the ’doc_root’ to point to your webservers document_root. ex: c:\apache\htdocs or c:\webroot
  - Choose which extensions you would like to load when PHP starts. You can uncomment the: ’extension=php_*.dll’ lines in `php.ini` to load these extensions. Some extensions require you to have additional libraries installed on your system for the module to work correctly. The PHP FAQ (http://www.php.net/FAQ.php) has more information on where to get supporting libraries. You can also load a module dynamically in your script using **dl()**. See the section about Windows extensions.
  - On PWS and IIS, you can set the `browscap.ini` to point to: ’c:\windows\system\inetsrv\browscap.ini’ on Windows 9x/Me and ’c:\winnt\system32\inetsrv\browscap.ini’ on NT/2000 Server. Additional information on using the browscap functionality in PHP can be found at this mirror (http://php.netvision.net.il/browser-id.php3), select the "source" button to see it in action.

Building from source

Before getting started, it is worthwhile answering the question: "Why is building on Windows so hard?" Two reasons come to mind:

1. Windows does not (yet) enjoy a large community of developers who are willing to freely share their source. As a direct result, the necessary investment in infrastructure required to support such development hasn’t been made. By and large, what is available has been made possible by the porting of necessary utilities from Unix. Don’t be surprised if some of this heritage shows through from time to time.
2. Pretty much all of the instructions that follow are of the "set and forget" variety. So sit back and try follow the instructions below as faithfully as you can.

Preparations

Before you get started, you have a lot to download....

- For starters, get the Cygwin toolkit from the closest cygwin (http://sources.redhat.com/cygwin/download.html) mirror site. This will provide you most of the popular GNU utilities used by the build process.
- Download the rest of the build tools you will need from the PHP site at http://www.php.net/extra/win32build.zip.
- Get the source code for the DNS name resolver used by PHP at http://www.php.net/extra/bindlib_w32.zip. This is a replacement for the `resolv.lib` library included in `win32build.zip`.
- If you don’t already have an unzip utility, you will need one. A free version is available from InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Finally, you are going to need the source to PHP 4 itself. You can get the latest development version using anonymous CVS (http://www.php.net/anoncvs.php). If you get a snapshot (http://snaps.php.net/) or a source (http://www.php.net/downloads.php) tarball, you not only will have to untar and ungzip it, but you will have to convert the bare linefeeds to crlf’s in the `*.dsp` and `*.dsw` files before Microsoft Visual C++ will have anything to do with them.

> **Poznámka:** Place the `Zend` and `TSRM` directories inside the `php4` directory in order for the projects to be found during the build process.

Putting it all together

- Follow the instructions for installing the unzip utility of your choosing.
- Execute `setup.exe` and follow the installation instructions. If you choose to install to a path other than `c:\cygnus`, let the build process know by setting the Cygwin environment variable. On Windows 95/98 setting an environment variable can be done by placing a line in your autoexec.bat. On Windows NT, go to My Computer => Control Panel => System and select the environment tab.

> **Varování**
>
> Make a temporary directory for Cygwin to use, otherwise many commands (particularly bison) will fail. On Windows 95/98, **`mkdir C:\TMP`**. For Windows NT, **`mkdir %SystemDrive%\tmp`**.

- Make a directory and unzip `win32build.zip` into it.
- Launch Microsoft Visual C++, and from the menu select Tools => Options. In the dialog, select the directories tab. Sequentially change the dropdown to Executables, Includes, and Library files, and ensure that `cygwin\bin`, `win32build\include`, and `win32build\lib` are in each list, respectively. (To add an entry, select a blank line at the end of the list and begin typing). Typical entries will look like this:
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Press OK, and exit out of Visual C++.

- Make another directory and unzip `bindlib_w32.zip` into it. Decide whether you want to have debug symbols available (bindlib - Win32 Debug) or not (bindlib - Win32 Release). Build the appropriate configuration:
  - For GUI users, launch VC++, and then select File => Open Workspace and select bindlib. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
  - For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**
  - At this point, you should have a usable `resolv.lib` in either your `Debug` or `Release` subdirectories. Copy this file into your `win32build\lib` directory over the file by the same name found in there.

Compiling

The best way to get started is to build the standalone/CGI version.

- For GUI users, launch VC++, and then select File => Open Workspace and select php4ts. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - At this point, you should have a usable `php.exe` in either your `Debug_TS` or `Release_TS` subdirectories.

Repeat the above steps with `php4isapi.dsp` (which can be found in `sapi\isapi`) in order to build the code necessary for integrating PHP with Microsoft IIS.

Installation of Windows extensions

After installing PHP and a webserver on Windows, you will probably want to install some extensions for added functionality. The following table describes some of the extensions available. As described in the manual installation steps, you can choose which extensions you would like to load when PHP starts by uncommenting the: 'extension=php_*.dll' lines in `php.ini`. Some extensions require you to have additional libraries installed on your system for the module to work correctly. The PHP FAQ (http://www.php.net/FAQ.php) has more information on where to get supporting libraries. You can also load a module dynamically in your script using **dl()**.

The DLLs for PHP extensions are prefixed with 'php_'. This prevents confusion between PHP extensions and their supporting libraries.

> **Poznámka:** In PHP 4.0.4pl1 MySQL, ODBC, FTP, Calendar, BCMath, COM, PCRE, Session, WDDX and XML support is *built-in*. You don't need to load any additional extensions in order to use these functions. See your distributions `README.txt` or `install.txt` for a list of built in modules.

Tabulka 2-1. PHP Extensions

php_calendar.dll	Calendar conversion functions
php_crypt.dll	Crypt functions
php_dbase.dll	dBase functions
php_dbm.dll	Berkeley DB2 library
php_filepro.dll	Read-only access to Filepro databases
php_gd.dll	GD library functions for GIF manipulation
php_hyperwave.dll	HyperWave functions
php_imap4r2.dll	IMAP 4 functions
php_ldap.dll	LDAP functions
php_msql1.dll	mSQL 1 client
php_msql2.dll	mSQL 2 client
php_mssql.dll	MSSQL client (requires MSSQL DB-Libraries
php3_mysql.dll (built into PHP 4)	MySQL functions
php_nsmail.dll	Netscape mail functions
php_oci73.dll	Oracle functions
php_snmp.dll	SNMP get and walk functions (NT only!)
php_zlib.dll	ZLib compression functions

Servers-Apache

This section contains notes and hints specific to Apache installs of PHP, both for Unix and Windows versions.

Details of installing PHP with Apache on Unix.

You can select arguments to add to the **configure** on line 8 below from the Complete list of configure options.

Příklad 2-5. Installation Instructions (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
```

```
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure -prefix=/www
7.  cd ../php-x.x.x
8.  ./configure -with-mysql -with-apache=../apache_1.3.x -enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. for PHP 3: ./configure -activate-module=src/modules/php3/libphp3.a
    for PHP 4: ./configure -activate-module=src/modules/php4/libphp4.a
13. make
14. make install

  Instead of this step you may prefer to simply copy the httpd binary
  overtop of your existing binary.  Make sure you shut down your
  server first though.

15. cd ../php-x.x.x
16. for PHP 3: cp php3.ini-dist /usr/local/lib/php3.ini
    for PHP 4: cp php.ini-dist /usr/local/lib/php.ini

  You can edit your .ini file to set PHP options.  If
  you prefer this file in another location, use
  -with-config-file-path=/path in step 8.

17. Edit your httpd.conf or srm.conf file and add:

     For PHP 3:   AddType application/x-httpd-php3 .php3
     For PHP 4:   AddType application/x-httpd-php .php

  You can choose any extension you wish here.  .php is simply the one
  we suggest. You can even include .html .

18. Use your normal procedure for starting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Depending on your Apache install and Unix variant, there are many possible ways to stop and restart the server. Below are some typical lines used in restarting the server, for different apache/unix installations. You should replace `/path/to/` with the path to these applications on your systems.

```
1. Several Linux and SysV variants:
/etc/rc.d/init.d/httpd restart

2. Using apachectl scripts:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl and httpsdctl (Using OpenSSL), similar to apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Using mod_ssl, or another SSL server, you may want to manually
stop and start:
/path/to/apachectl stop
/path/to/apachectl startssl
```

The locations of the apachectl and http(s)dctl binaries often vary. If your system has `locate` or `whereis` or `which` commands, these can assist you in finding your server contrl programs.

Different examples of compiling PHP for apache are as follows:

```
./configure -with-apxs -with-pgsql
```

This will create a `libphp4.so` shared library that is loaded into Apache using a LoadModule line in Apache's `httpd.conf` file. The PostgreSQL support is embedded into this `libphp4.so` library.

```
./configure -with-apxs -with-pgsql=shared
```

This will again create a `libphp4.so` shared library for Apache, but it will also create a `pgsql.so` shared library that is loaded into PHP either by using the extension directive in `php.ini` file or by loading it explicitly in a script using the **dl()** function.

```
./configure -with-apache=/path/to/apache_source -with-pgsql
```

This will create a `libmodphp4.a` library, a `mod_php4.c` and some accompanying files and copy this into the `src/modules/php4` directory in the Apache source tree. Then you compile Apache using `-activate-module=src/modules/php4/libphp4.a` and the Apache build system will create `libphp4.a` and link it statically into the `httpd` binary. The PostgreSQL support is included directly into this `httpd` binary, so the final result here is a single `httpd` binary that includes all of Apache and all of PHP.

```
./configure -with-apache=/path/to/apache_source -with-pgsql=shared
```

Same as before, except instead of including PostgreSQL support directly into the final `httpd` you will get a `pgsql.so` shared library that you can load into PHP from either the `php.ini` file or directly using **dl()**.

When choosing to build PHP in different ways, you should consider the advantages and drawbacks of each method. Building as a shared object will mean that you can compile apache separately, and don't have to recompile everything as you add to, or change, PHP. Building PHP into apache (static method) means that PHP will load and run faster. For more information, see the Apache webpage on DSO support (http://www.apache.org/docs/dso.html).

Details of installing PHP on Windows with Apache 1.3.x

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (php.exe), the other is to use the Apache module dll. In either case you need to stop the Apache server, and edit your `srm.conf` or `httpd.conf` to configure Apache to work with PHP.

Although there can be a few variations of configuring PHP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Docs for further configuration directives.

If you unziped the PHP package to C:\PHP\ as desribed in the General Installation Steps section, you need to insert these lines to your Apache conf file to set up the CGI binary:

- `ScriptAlias /php/ "c:/php/"`
- `AddType application/x-httpd-php .php .phtml`
- `Action application/x-httpd-php "/php/php.exe"`

Remerber to restart the server, for example, `NET STOP APACHE` followed by `NET START APACHE`.

If you would like to use PHP as a module in Apache, you should move `php4ts.dll` to the windows/system (for Windows 9x/Me) or winnt/system32 (for Windows NT/2000) directory, overwriting any older file. Then you should add the following two lines to you Apache conf file:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

To use the source code highlighting feature, simply create a PHP script file and stick this code in: `<?php show_source ("original_php_script.php"); ?>`. Substitute `original_php_script.php` with the name of the file you wish to show the source of. (This is the only way of doing so).

> **Poznámka:** On Win-Apache all backslashes in a path statement such as: "c:\directory\file.ext", must be converted to forward slashes.

Servers-CGI/Commandline

The default is to build PHP as a CGI program. This creates a commandline interpreter, which can be used for CGI processing, or for non-web-related PHP scripting. If you are running a web server PHP has module support for, you should generally go for that solution for performance reasons. However, the CGI version enables Apache users to run different PHP-enabled pages under different user-ids. Please make sure you read through the Security chapter if you are going to run PHP as a CGI.

Testing

If you have built PHP as a CGI program, you may test your build by typing **make test**. It is always a good idea to test your build. This way you may catch a problem with PHP on your platform early instead of having to struggle with it later.

Benchmarking

If you have built PHP 3 as a CGI program, you may benchmark your build by typing **make bench**. Note that if safe mode is on by default, the benchmark may not be able to finish if it takes longer then the 30 seconds allowed. This is because the **set_time_limit()** can not be used in safe mode. Use the max_execution_time configuration setting to control this time for your own scripts. **make bench** ignores the configuration file.

> **Poznámka: make bench** is only available for PHP 3.

Servers-fhttpd

To build PHP as an fhttpd module, answer "yes" to "Build as an fhttpd module?" (the `-with-fhttpd=DIR` option to configure) and specify the fhttpd source base directory. The default directory is `/usr/local/src/fhttpd`. If you are running fhttpd, building PHP as a module will give better performance, more control and remote execution capability.

Servers-Caudium

PHP 4 can be build as a Pike module for the Caudium webserver. Note that this is not supported with PHP 3. Follow the simple instructions below to install PHP 4 for Caudium.

Příklad 2-6. Caudium Installation Instructions

```
1.  Make sure you have Caudium installed prior to attempting to
    install PHP 4. For PHP 4 to work correctly, you will need Pike
    7.0.268 or newer. For the sake of this example we assume that
    Caudium is installed in /opt/caudium/server/.
2.  Change directory to php-x.y.z (where x.y.z is the version number).
3.  ./configure -with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Restart Caudium if it's currently running.
7.  Log into the graphical configuration interface and go to the
    virtual server where you want to add PHP 4 support.
```

```
8.  Click Add Module and locate and then add the PHP 4 Script Support module.
9.  If the documentation says that the 'PHP 4 interpreter isn't
    available', make sure that you restarted the server. If you did
    check /opt/caudium/logs/debug/default.1 for any errors related to
    PHP4.so. Also make sure that
    caudium/server/lib/[pike-version]/PHP4.so
    is present.
10. Configure the PHP Script Support module if needed.
```

You can of course compile your Caudium module with support for the various extensions available in PHP 4. See the complete list of configure options for an exhaustive rundown.

> **Poznámka:** When compiling PHP 4 with MySQL support you must make sure that the normal MySQL client code is used. Otherwise there might be conflicts if your Pike already has MySQL support. You do this by specifying a MySQL install directory the –with-mysql option.

Servers-IIS/PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server) installing PHP for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the INF file included with the distribution (php_iis_reg.inf). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

> **Varování**
>
> These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- On the edit menu select: `New->String Value`.
- Type in the extension you wish to use for your php scripts. ex: `.php`
- Double click on the new string value and enter the path to `php.exe` in the value data field. ex: `c:\php\php.exe %s %s`. The '%s %s' is VERY important, PHP will not work properly without it.
- Repeat these steps for each extension you wish to associate with PHP scripts.
- Now navigate to: `HKEY_CLASSES_ROOT`
- On the edit menu select: `New->Key`.
- Name the key to the extension you setup in the previous section. ex: `.php`
- Highlight the new key and in the right side pane, double click the "default value" and enter `phpfile`.
- Repeat the last step for each extension you set up in the previous section.
- Now create another `New->Key` under `HKEY_CLASSES_ROOT` and name it `phpfile`.
- Highlight the new key `phpfile` and in the right side pane, double click the "default value" and enter `PHP Script`.
- Right click on the `phpfile` key and select `New->Key`, name it `Shell`.
- Right click on the `Shell` key and select `New->Key`, name it `open`.
- Right click on the `open` key and select `New->Key`, name it `command`.

- Highlight the new key `command` and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: `c:\php\php.exe -q %1`. (don't forget the `%1`).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool (http://www.genusa.com/iis/iiscfg.html) from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module dll.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` file (look into the sapi dir) to reflect the location of your php.exe. Forward slashes should be escaped, for example:
```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map]
".php"="C:\\PHP\\php.exe"
```
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` file (look into the sapi dir) to reflect the location of your php4isapi.dll. Forward slashes should be escaped, for example:
```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map]
".php"="C:\\PHP\\sapi\\php4isapi.dll"
```
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000 and IIS 4 or newer

To install PHP on an NT/2000 Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (php.exe) or with the ISAPI module.

In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.
- Click Add, and in the Executable box, type: `c:\php\php.exe %s %s` (assuming that you have unziped PHP in c:\php\). You MUST have the %s %s on the end, PHP will not function properly if you fail to do this.
- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the Script engine checkbox. You must repeat step 3 and 4 for each extension you want associated with PHP scripts. (`.php` and `.phtml` are common.)
- Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for I_USR_ to the directory that contains `php.exe`.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the php4isapi.dll.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the php4isapi.dll as the Executable, supply .php as the extension, leave Method exclusions blank, and check the Script engine checkbox.
- Stop IIS completely
- Start IIS again

Servers-Netscape and iPlanet

To build PHP with NES or iPlanet web servers, enter the proper install directory for the `-with-nsapi = DIR` option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

Příklad 2-7. Installation Example for Netscape Enterprise on Solaris

```
Instructions for Sun Solaris 2.6 with Netscape Enterprise Server 3.6
From: bhager@invacare.com

1. Install the following packages from www.sunfreeware.com or another
download site:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)

2. Make sure your path includes the proper directories
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (if you have a .gz dist, otherwise go to 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. For the following step, make sure /opt/netscape/suitespot/ is where
   your netscape server is installed. Otherwise, change to correct path:
    /configure -with-mysql=/usr/local/mysql -with-nsapi=/opt/netscape/suitespot/ -
enable-track-vars -enable-libgcc
7. make
8. make install
```

After performing the base install and reading the appropriate readme file, you may need to performs some additional configuration steps.

Firstly you may need to add some paths to the LD_LIBRARY_PATH environment for Netscape to find all the shared libs. This can best done in the start script for your Netscape server. Windows users can probably skip this step. The start script is often located in: `/path/to/server/https-servername/start`

You may also need to edit the configuration files that are located in:`/path/to/server/https-servername/config/`.

Příklad 2-8. Configuration Example for Netscape Enterprise

```
Configuration Instructions for Netscape Enterprise Server
From: bhager@invacare.com

1. Add the following line to mime.types:
    type=magnus-internal/x-httpd-php exts=php

2. Add the following to obj.conf, shlib will vary depending on
    your OS, for unix it will be something like
    /opt/netscape/suitespot/bin/libphp4.so.

    You should place the following lines after mime types init.
    Init fn="load-
modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib="/php4/nsapiPHP4.dll"
    Init fn=php4_init errorString="Failed to initialize PHP!"

    <object name="default">
    .
    .
    .
    .#NOTE this next line should happen after all 'ObjectType' and be-
fore all 'AddLog' lines
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Authentication configuration

    PHP authentication cannot be used with any other authentication. ALL AUTHENTICA-
TION IS
    PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the en-
tire server, add
    the following line:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    To use PHP Authentication on a single directory, add the following:

    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

Servers-OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd.

OmniHTTPd 2.0b1 and up for Windows

This has got to be the easiest config there is:

- Step 1: Install OmniHTTPd server.
- Step 2: Right click on the blue OmniHTTPd icon in the system tray and select `Properties`
- Step 3: Click on `Web Server Global Settings`
- Step 4: On the 'External' tab, enter: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, and use the Add button.
- Step 5: On the `Mime` tab, enter: `virtual = wwwserver/stdcgi | actual = .php`, and use the Add button.
- Step 6: Click `OK`

Repeat steps 2 - 6 for each extension you want to associate with PHP.

> **Poznámka:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Servers-Oreilly Website Pro

This section contains notes and hints specific to Oreilly Website Pro.

Oreilly Website Pro 2.5 and up for Windows

This list describes how to set up the PHP CGI binary or the ISAPI module to work with Oreilly Website Pro on Windows.

- Edit the Server Properties and select the tab "Mapping".
- From the List select "Associations" and enter the desired extension (".php") and the path to the CGI exe (ex. c:\php\php.exe) or the ISAPI dll file (ex. c:\php\sapi\php4isapi.dll).
- Select "Content Types" add the same extension ".php" and enter the content type. If you choose the CGI exe file, enter 'wwwserver/shellcgi', if you chosse the ISAPI module, enter 'wwwserver/isapi' (both without quotes).

Servers-Xitami

This section contains notes and hints specific to Xitami.

Xitami for Windows

This list describes how to set up the PHP CGI binary to work with Xitami on Windows.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually http://127.0.0.1/admin), and click on Configuration.
- Navigate to the Filters, and put the extension which php should parse (i.e. .php) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your php executable i.e. c:\php\php.exe.
- Press the 'Save' icon.

Servers-Other web servers

PHP can be built to support a large number of web servers. Please see Server-related options for a full list of server-related configure options. The PHP CGI binaries are compatible with almost all webservers supporting the CGI interface.

Problems?

Read the FAQ

Some problems are more common than others. The most common ones are listed in the PHP FAQ, found at http://www.php.net/FAQ.php

Other problems

If you are still stuck, someone on the PHP installation mailing list may be able to help you. You should check out the archive first, in case someone already answered someone else who had the same problem as you. The archives are available from the support page on http://www.php.net/. To subscribe to the PHP installation mailing list, send an empty mail to php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). The mailing list address is `php-install@lists.php.net`.

If you want to get help on the mailing list, please try to be precise and give the necessary details about your environment (which operating system, what PHP version, what web server, if you are running PHP as CGI or a server module, etc.), and preferably enough code to make others able to reproduce and test your problem.

Bug reports

If you think you have found a bug in PHP, please report it. The PHP developers probably don’t know about it, and unless you report it, chances are it won’t be fixed. You can report bugs using the bug-tracking system at http://bugs.php.net/.

Read the Bugs-Dos-And-Donts (http://bugs.php.net/bugs-dos-and-donts.php) before submitting any bug reports!

Kapitola 3. Configuration

The configuration file

The configuration file (called `php3.ini` in PHP 3.0, and simply `php.ini` as of PHP 4.0) is read when PHP starts up. For the server module versions of PHP, this happens only once when the web server is started. For the CGI version, it happens on every invocation.

When using PHP as an Apache module, you can also change the configuration settings using directives in Apache configuration files and .htaccess files.

With PHP 3.0, there are Apache directives that correspond to each configuration setting in the `php3.ini` name, except the name is prefixed by "php3_".

With PHP 4.0, there are just a few Apache directives that allow you to change the PHP configuration settings.

php_value *`name value`*

This sets the value of the specified variable.

php_flag *`name on|off`*

This is used to set a Boolean configuration option.

php_admin_value *`name value`*

This sets the value of the specified variable. "Admin" configuration settings can only be set from within the main Apache configuration files, and not from .htaccess files.

php_admin_flag *`name on|off`*

This is used to set a Boolean configuration option.

You can view the settings of the configuration values in the output of **phpinfo()**. You can also access the values of individial configuration settings using **get_cfg_var()**.

General Configuration Directives

`allow_url_fopen` boolean

This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.

Poznámka: This option was introduced immediately after the release of version 4.0.3. For versions up to and including 4.0.3 you can only disable this feature at compile time by using the configuration switch *`-disable-url-fopen-wrapper`*.

`asp_tags` boolean

Enables the use of ASP-like <% %> tags in addition to the usual <?php ?> tags. This includes the variable-value printing shorthand of <%= $value %>. For more information, see Escaping from HTML.

Poznámka: Support for ASP-style tags was added in 3.0.4.

`auto_append_file` string

Specifies the name of a file that is automatically parsed after the main file. The file is included as if it was called with the **include()** function, so include_path is used.

The special value none disables auto-appending.

Poznámka: If the script is terminated with **exit()**, auto-append will *not* occur.

`auto_prepend_file` string

Specifies the name of a file that is automatically parsed before the main file. The file is included as if it was called with the **include()** function, so include_path is used.

The special value none disables auto-prepending.

`cgi_ext` string

`display_errors` boolean

This determines whether errors should be printed to the screen as part of the HTML output or not.

`doc_root` string

PHP's "root directory" on the server. Only used if non-empty. If PHP is configured with safe mode, no files outside this directory are served.

`engine` boolean

This directive is really only useful in the Apache module version of PHP. It is used by sites that would like to turn PHP parsing on and off on a per-directory or per-virtual server basis. By putting **`engine off`** in the appropriate places in the `httpd.conf` file, PHP can be enabled or disabled.

`error_log` string

Name of file where script errors should be logged. If the special value `syslog` is used, the errors are sent to the system logger instead. On UNIX, this means syslog(3) and on Windows NT it means the event log. The system logger is not supported on Windows 95.

`error_reporting` integer

Set the error reporting level. The parameter is an integer representing a bit field. Add the values of the error reporting levels you want.

Tabulka 3-1. Error Reporting Levels

bit value	enabled reporting
1	normal errors
2	normal warnings
4	parser errors
8	non-critical style-related warnings

The default value for this directive is 7 (normal errors, normal warnings and parser errors are shown).

`open_basedir` string

Limit the files that can be opened by PHP to the specified directory-tree.

When a script tries to open a file with, for example, fopen or gzopen, the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. All symbolic links are resolved, so it's not possible to avoid this restriction with a symlink.

The special value . indicates that the directory in which the script is stored will be used as base-directory.

Under Windows, separate the directories with a semicolon. On all other systems, separate the directories with a colon. As an Apache module, open_basedir paths from parent directories are now automatically inherited.

Poznámka: Support for multiple directories was added in 3.0.7.

The default is to allow all files to be opened.

gpc_order string

Set the order of GET/POST/COOKIE variable parsing. The default setting of this directive is "GPC". Setting this to "GP", for example, will cause PHP to completely ignore cookies and to overwrite any GET method variables with POST-method variables of the same name.

ignore_user_abort string

On by default. If changed to Off scripts will be terminated as soon as they try to output something after a client has aborted their connection. **ignore_user_abort()**.

include_path string

Specifies a list of directories where the **require()**, **include()** and **fopen_with_path()** functions look for files. The format is like the system's PATH environment variable: a list of directories separated with a colon in UNIX or semicolon in Windows.

Příklad 3-1. UNIX include_path

```
include_path=.:/home/httpd/php-lib
```

Příklad 3-2. Windows include_path

```
include_path=".;c:\www\phplib"
```

The default value for this directive is `.` (only the current directory).

isapi_ext string

log_errors boolean

Tells whether script error messages should be logged to the server's error log. This option is thus server-specific.

magic_quotes_gpc boolean

Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are on, all ' (single-quote), " (double quote), \ (backslash) and NUL's are escaped with a backslash automatically. If magic_quotes_sybase is also on, a single-quote is escaped with a single-quote instead of a backslash.

magic_quotes_runtime boolean

If *magic_quotes_runtime* is enabled, most functions that return data from any sort of external source including databases and text files will have quotes escaped with a backslash. If *magic_quotes_sybase* is also on, a single-quote is escaped with a single-quote instead of a backslash.

magic_quotes_sybase boolean

If *magic_quotes_sybase* is also on, a single-quote is escaped with a single-quote instead of a backslash if *magic_quotes_gpc* or *magic_quotes_runtime* is enabled.

max_execution_time integer

This sets the maximum time in seconds a script is allowed to take before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The default setting is `30`.

memory_limit integer

This sets the maximum amount of memory in bytes that a script is allowed to allocate. This helps prevent poorly written scripts for eating up all available memory on a server.

nsapi_ext string

register_globals boolean

Tells whether or not to register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. You may want to turn this off if you don't want to clutter your scripts' global scope with user data. This makes the most sense when coupled with track_vars - in which case you can access all of the EGPCS variables through the `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS`, and `$HTTP_SERVER_VARS` arrays in the global scope.

short_open_tag boolean

Tells whether the short form (**<? ?>**) of PHP's open tag should be allowed. If you want to use PHP in combination with XML, you have to disable this option. If disabled, you must use the long form of the open tag (**<?php ?>**).

sql.safe_mode boolean

track_errors boolean

If enabled, the last error message will always be present in the global variable $php_errormsg.

track_vars boolean

If enabled, then Environment, GET, POST, Cookie, and Server variables can be found in the global associative arrays `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS`, and `$HTTP_SERVER_VARS`.

Note that as of PHP 4.0.3, track_vars is always turned on.

upload_tmp_dir string

The temporary directory used for storing files when doing file upload. Must be writable by whatever user PHP is running as.

user_dir string

The base name of the directory used on a user's home directory for PHP files, for example `public_html`.

warn_plus_overloading boolean

If enabled, this option makes PHP output a warning when the plus (+) operator is used on strings. This is to make it easier to find scripts that need to be rewritten to using the string concatenator instead (`.`).

Mail Configuration Directives

SMTP string

DNS name or IP address of the SMTP server PHP under Windows should use for mail sent with the **mail()** function.

sendmail_from string

Which "From:" mail address should be used in mail sent from PHP under Windows.

sendmail_path string

Where the **sendmail** program can be found, usually `/usr/sbin/sendmail` or `/usr/lib/sendmail` **configure** does an honest attempt of locating this one for you and set a default, but if it fails, you can set it here.

Systems not using sendmail should set this directive to the sendmail wrapper/replacement their mail system offers, if any. For example, Qmail (http://www.qmail.org/) users can normally set it to `/var/qmail/bin/sendmail`.

Safe Mode Configuration Directives

safe_mode boolean

Whether to enable PHP's safe mode. Read the Security chapter for more more information.

safe_mode_exec_dir string

If PHP is used in safe mode, **system()** and the other functions executing system programs refuse to start programs that are not in this directory.

Debugger Configuration Directives

debugger.host string

DNS name or IP address of host used by the debugger.

debugger.port string

Port number used by the debugger.

debugger.enabled boolean

Whether the debugger is enabled.

Extension Loading Directives

enable_dl boolean

This directive is really only useful in the Apache module version of PHP. You can turn dynamic loading of PHP extensions with **dl()** on and off per virtual server or per directory.

The main reason for turning dynamic loading off is security. With dynamic loading, it's possible to ignore all the safe_mode and open_basedir restrictions.

The default is to allow dynamic loading, except when using safe-mode. In safe-mode, it's always imposible to use **dl()**.

extension_dir string

In what directory PHP should look for dynamically loadable extensions.

extension string

Which dynamically loadable extensions to load when PHP starts up.

MySQL Configuration Directives

mysql.allow_persistent boolean

Whether to allow persistent MySQL connections.

mysql.default_host string

The default server host to use when connecting to the database server if no other host is specified.

mysql.default_user string

The default user name to use when connecting to the database server if no other name is specified.

mysql.default_password string

The default password to use when connecting to the database server if no other password is specified.

mysql.max_persistent integer

The maximum number of persistent MySQL connections per process.

mysql.max_links integer

The maximum number of MySQL connections per process, including persistent connections.

mSQL Configuration Directives

msql.allow_persistent boolean

Whether to allow persistent mSQL connections.

msql.max_persistent integer

The maximum number of persistent mSQL connections per process.

msql.max_links integer

The maximum number of mSQL connections per process, including persistent connections.

Postgres Configuration Directives

pgsql.allow_persistent boolean

Whether to allow persistent Postgres connections.

pgsql.max_persistent integer

The maximum number of persistent Postgres connections per process.

pgsql.max_links integer

The maximum number of Postgres connections per process, including persistent connections.

SESAM Configuration Directives

sesam_oml string

Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. The BS2000 PLAM library must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

sesam_configfile string

Name of SESAM application configuration file. Required for using SESAM functions. The BS2000 file must be readable by the apache server's user id.

The application configuration file will usually contain a configuration like (see SESAM reference manual):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive.

The message catalog must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

Sybase Configuration Directives

sybase.allow_persistent boolean

Whether to allow persistent Sybase connections.

sybase.max_persistent integer

The maximum number of persistent Sybase connections per process.

sybase.max_links integer

The maximum number of Sybase connections per process, including persistent connections.

Sybase-CT Configuration Directives

sybct.allow_persistent boolean

Whether to allow persistent Sybase-CT connections. The default is on.

sybct.max_persistent integer

The maximum number of persistent Sybase-CT connections per process. The default is -1 meaning unlimited.

sybct.max_links integer

The maximum number of Sybase-CT connections per process, including persistent connections. The default is -1 meaning unlimited.

sybct.min_server_severity integer

Server messages with severity greater than or equal to sybct.min_server_severity will be reported as warnings. This value can also be set from a script by calling **sybase_min_server_severity()**. The default is 10 which reports errors of information severity or greater.

sybct.min_client_severity integer

Client library messages with severity greater than or equal to sybct.min_client_severity will be reported as warnings. This value can also be set from a script by calling **sybase_min_client_severity()**. The default is 10 which effectively disables reporting.

sybct.login_timeout integer

The maximum time in seconds to wait for a connection attempt to succeed before returning failure. Note that if max_execution_time has been exceeded when a connection attempt times out, your script will be terminated before it can take action on failure. The default is one minute.

sybct.timeout integer

The maximum time in seconds to wait for a select_db or query operation to succeed before returning failure. Note that if max_execution_time has been exceeded when am operation times out, your script will be terminated before it can take action on failure. The default is no limit.

sybct.hostname string

The name of the host you claim to be connecting from, for display by sp_who. The default is none.

Informix Configuration Directives

ifx.allow_persistent boolean

Whether to allow persistent Informix connections.

ifx.max_persistent integer

The maximum number of persistent Informix connections per process.

ifx.max_links integer

The maximum number of Informix connections per process, including persistent connections.

ifx.default_host string

The default host to connect to when no host is specified in **ifx_connect()** or **ifx_pconnect()**.

ifx.default_user string

The default user id to use when none is specified in **ifx_connect()** or **ifx_pconnect()**.

ifx.default_password string

The default password to use when none is specified in **ifx_connect()** or **ifx_pconnect()**.

ifx.blobinfile boolean

Set to true if you want to return blob columns in a file, false if you want them in memory. You can override the setting at runtime with **ifx_blobinfile_mode()**.

`ifx.textasvarchar` boolean

Set to true if you want to return TEXT columns as normal strings in select statements, false if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

`ifx.byteasvarchar` boolean

Set to true if you want to return BYTE columns as normal strings in select queries, false if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

`ifx.charasvarchar` boolean

Set to true if you want to trim trailing spaces from CHAR columns when fetching them.

`ifx.nullformat` boolean

Set to true if you want to return NULL columns as the literal string "NULL", false if you want them returned as the empty string "". You can override this setting at runtime with **ifx_nullformat()**.

BC Math Configuration Directives

`bcmath.scale` integer

Number of decimal digits for all bcmath functions.

Browser Capability Configuration Directives

`browscap` string

Name of browser capabilities file. See also **get_browser()**.

Unified ODBC Configuration Directives

`uodbc.default_db` string

ODBC data source to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

`uodbc.default_user` string

User name to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

`uodbc.default_pw` string

Password to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

`uodbc.allow_persistent` boolean

Whether to allow persistent ODBC connections.

`uodbc.max_persistent` integer

The maximum number of persistent ODBC connections per process.

`uodbc.max_links` integer

The maximum number of ODBC connections per process, including persistent connections.

Kapitola 4. Security

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need.

As there are many different ways of utilizing PHP, there are many configuration options controlling its behaviour. A large selection of options guarantees you can use PHP for a lot of purposes, but it also means there are combinations of these options and server configurations that result in an insecure setup.

The configuration flexibility of PHP is equally rivalled by the code flexibility. PHP can be used to build complete server applications, with all the power of a shell user, or it can be used for simple server-side includes with little risk in a tightly controlled environment. How you build that environment, and how secure it is, is largely up to the PHP developer.

This chapter starts by explaining the different configuration option combinations and the situations they can be safely used. It then describes different considerations in coding for different levels of security, and ends with some general security advice.

Installed as CGI binary

Possible attacks

Using PHP as a CGI binary is an option for setups that for some reason do not wish to integrate PHP as a module into server software (like Apache), or will use PHP with different kinds of CGI wrappers to create safe chroot and setuid environments for scripts. This setup usually involves installing executable PHP binary to the web server cgi-bin directory. CERT advisory CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recommends against placing any interpreters into cgi-bin. Even if the PHP binary can be used as a standalone interpreter, PHP is designed to prevent the attacks this setup makes possible:

- Accessing system files: `http://my.host/cgi-bin/php?/etc/passwd`

  The query information in a url after the question mark (?) is passed as command line arguments to the interpreter by the CGI interface. Usually interpreters open and execute the file specified as the first argument on the command line.

  When invoked as a CGI binary, PHP refuses to interpret the command line arguments.

- Accessing any web document on server: `http://my.host/cgi-bin/php/secret/doc.html`

  The path information part of the url after the PHP binary name, `/secret/doc.html` is conventionally used to specify the name of the file to be opened and interpreted by the CGI program. Usually some web server configuration directives (Apache: Action) are used to redirect requests to documents like `http://my.host/secret/script.php` to the PHP interpreter. With this setup, the web server first checks the access permissions to the directory `/secret`, and after that creates the redirected request `http://my.host/cgi-bin/php/secret/script.php`. Unfortunately, if the request is originally given in this form, no access checks are made by web server for file `/secret/script.php`, but only for the `/cgi-bin/php` file. This way any user able to access `/cgi-bin/php` is able to access any protected document on the web server.

  In PHP, compile-time configuration option –enable-force-cgi-redirect and runtime configuration directives doc_root and user_dir can be used to prevent this attack, if the server document tree has any directories with access restrictions. See below for full the explanation of the different combinations.

Case 1: only public files served

If your server does not have any content that is not restricted by password or ip based access control, there is no need for these configuration options. If your web server does not allow you to do redirects, or the server does not have a way to communicate to the PHP binary that the request is a safely redirected request, you can specify the option –enable-force-cgi-redirect to the configure script. You still have to make sure your PHP scripts do not rely on one or another way of calling the script, neither by directly `http://my.host/cgi-bin/php/dir/script.php` nor by redirection `http://my.host/dir/script.php`.

Redirection can be configured in Apache by using AddHandler and Action directives (see below).

Case 2: using –enable-force-cgi-redirect

This compile-time option prevents anyone from calling PHP directly with a url like `http://my.host/cgi-bin/php/secretdir/script.php`. Instead, PHP will only parse in this mode if it has gone through a web server redirect rule.

Usually the redirection in the Apache configuration is done with the following directives:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

This option has only been tested with the Apache web server, and relies on Apache to set the non-standard CGI environment variable REDIRECT_STATUS on redirected requests. If your web server does not support any way of telling if the request is direct or redirected, you cannot use this option and you must use one of the other ways of running the CGI version documented here.

Case 3: setting doc_root or user_dir

To include active content, like scripts and executables, in the web server document directories is sometimes consider an insecure practice. If, because of some configuration mistake, the scripts are not executed but displayed as regular HTML documents, this may result in leakage of intellectual property or security information like passwords. Therefore many sysadmins will prefer setting up another directory structure for scripts that are accessible only through the PHP CGI, and therefore always interpreted and not displayed as such.

Also if the method for making sure the requests are not redirected, as described in the previous section, is not available, it is necessary to set up a script doc_root that is different from web document root.

You can set the PHP script document root by the configuration directive doc_root in the configuration file, or you can set the environment variable PHP_DOCUMENT_ROOT. If it is set, the CGI version of PHP will always construct the file name to open with this *`doc_root`* and the path information in the request, so you can be sure no script is executed outside this directory (except for *`user_dir`* below).

Another option usable here is user_dir. When user_dir is unset, only thing controlling the opened file name is *`doc_root`*. Opening an url like `http://my.host/~user/doc.php` does not result in opening a file under users home directory, but a file called `~user/doc.php` under doc_root (yes, a directory name starting with a tilde [~]).

If user_dir is set to for example `public_php`, a request like `http://my.host/~user/doc.php` will open a file called `doc.php` under the directory named `public_php` under the home directory of the user. If the home of the user is `/home/user`, the file executed is `/home/user/public_php/doc.php`.

`user_dir` expansion happens regardless of the *`doc_root`* setting, so you can control the document root and user directory access separately.

Case 4: PHP parser outside of web tree

A very secure option is to put the PHP parser binary somewhere outside of the web tree of files. In `/usr/local/bin`, for example. The only real downside to this option is that you will now have to put a line similar to:

```
#!/usr/local/bin/php
```

as the first line of any file containing PHP tags. You will also need to make the file executable. That is, treat it exactly as you would treat any other CGI script written in Perl or sh or any other common scripting language which uses the `#!` shell-escape mechanism for launching itself.

To get PHP to handle PATH_INFO and PATH_TRANSLATED information correctly with this setup, the php parser should be compiled with the –enable-discard-path configure option.

Installed as an Apache module

When PHP is used as an Apache module it inherits Apache's user permissions (typically those of the "nobody" user). This has several impacts on security and authorization. For example, if you are using PHP to access a database, unless that database has built-in access control, you will have to make the database accessable to the "nobody" user. This means a malicious script could access and modify the databse, even without a username and password. It's entirely possible that a web spider could stumble across a database adminisitror's web page, and drop all of your databases. You can protect against this with Apache authorization, or you can design your own access model using LDAP, .htaccess files, etc. and include that code as part of your PHP scripts.

Often, once security is established to the point where the PHP user (in this case, the apache user) has very little risk, it is discovered that PHP now has been prevented from writing virus files to user directories. Or perhaps it has been prevented from accessing or changing a non-public database. It has equally been secured from writing files that it should, or entering database transactions.

A frequent security mistake made at this point is to allow apache root permissions.

Escalating the Apache user's permissions to root is extremely dangerous and may compromise the entire system, so sudo'ing, chroot'ing ,or otherwise running as root should not be considered by those who are not security professionals.

Filesystem Security

PHP is subject to the security built into most server systems with respect to permissions on a file and directory basis. This allows you to control which files in the filesystem may be read. Care should be taken with any files which are world readable to ensure that they are safe for reading by all users who have access to that filesystem.

Since PHP was designed to allow user level access to the filesystem, it's entirely possible to write a PHP script that will allow you to read system files such as /etc/password, modify your ethernet connections, send massive printer jobs out, etc. This has some obvious implications, in that you need to ensure that the files that you read from and write to are the appropriate ones.

Consider the following script, where a user indicates that they'd like to delete a file in their home directory. This assumes a situation where a PHP web interface is regularly used for file management, so the Apache user is allowed to delete files in the user home directories.

Příklad 4-1. Poor variable checking leads to....

```
<?php
// remove a file from the user's home directory
$username = $user_submitted_name;
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ($homedir/$userfile);
echo "$file_to_delete has been deleted!";
?>
```

Since the username is postable from a user form, they can submit a username and file belonging to someone else, and delete files. In this case, you'd want to use some other form of authentication. Consider what could happen if the variables submitted were "../etc/" and "passwd". The code would then effectively read:

Příklad 4-2. ... A filesystem attack

```
<?php
// removes a file from anywhere on the hard drive that
// the PHP user has access to. If PHP has root access:
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd has been deleted!";
?>
```

There are two important measures you should take to prevent these issues.

- Only allow limited permissions to the PHP web user binary.
- Check all variables which are submitted.

Here is an improved script:

Příklad 4-3. More secure file name checking

```
<?php
// removes a file from the hard drive that
// the PHP user has access to.
$username = $HTTP_REMOTE_USER; // use an authentication mechanisim

$homedir = "/home/$username";

$file_to_delete = basename("$userfile"); // strip paths
unlink ($homedir/$file_to_delete);

$fp = fopen("/home/logging/filedelete.log","+a"); //log the deletion
$logstring = "$HTTP_REMOTE_USER $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);

echo "$file_to_delete has been deleted!";
?>
```

Alternately, you may prefer to write a more customized check:

Příklad 4-4. More secure file name checking

```
<?php
$username = getenv("REMOTE_USER");
$homedir = "/home/$username";

if (!ereg('^[^./][^/]*$', $userfile))
    die('bad filename'); //die, do not process

//etc...
?>
```

Depending on your operating system, there are a wide variety of files which you should be concerned about, including device entries (/dev/ or COM1), configuration files (/etc/ files and the .ini files), well known file storage areas (/home/, My Documents), etc. For this reason, it's usually easier to create a policy where you forbid everything except for what you explicitly allow.

Error Reporting

A standard attack tactic involves profiling a system by feeding it improper data, and checking for the kinds, and contexts, of the errors which are returned. This allows the system cracker to probe for information about the server, to determine possible weaknesses.

The PHP errors which are normally returned can be quite helpful to a developer who is trying to debug a script, indicating such things as the function or file that failed, the PHP file it failed in, and the line number which the failure occured in. This is all information that can be exploited. It is not uncommon for a php developer to use **show_source()**, **highlight_string()**, or **highlight_file()** as a debugging measure, but in a live site, this can expose hidden variables, unchecked syntax, and other dangerous information.

For example, the very style of a generic error indicates a system is running PHP. If the attacker was looking at an .html page, and wanted to probe for the back-end (to look for known weaknesses in the system), by feeding it the wrong data they may be able to determine that a system was built with PHP.

A function error can indicate whether a system may be running a specific database engine, or give clues as to how a web page or programmed or designed. This allows for deeper investigation into open database ports, or to look for specific bugs or weaknesses in a web page. By feeding different pieces of bad data, for example, an attacker can determine the order of authentication in a script, (from the line number errors) as well as probe for exploits that may be exploited in different locations in the script.

A filesystem or general PHP error can indicate what permissions the webserver has, as well as the structure and organization of files on the web server. Developer written error code can aggravate this problem, leading to easy exploitation of formerly "hidden" information.

There are three major solutions to this issue. The first is to scrutinize all functions, and attempt to compensate for the bulk of the errors. The second is to disable error reporting entirely on the running code. The third is to use PHP's custom error handling functions to create your own error handler. Depending on your security policy, you may find all three to be applicable to your situation.

User Submitted Data

The greatest weakness in many PHP programs is not inherent in the language itself, but merely an issue of code not being written with security in mind. For this reason, you should always take the time to consider the implications of a given piece of code, to ascertain the possible damage if an unexpected variable is submitted to it.

Příklad 4-5. Dangerous Variable Usage

```
<?php
// remove a file from the user's home directory... or maybe
// somebody else's?
unlink ($evil_var);

// Write logging of their access... or maybe not?
fputs ($fp, $evil_var);

// Execute something trivial.. or rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

You should always carefully examine your code to make sure that any variables being submitted from a web browser are being properly checked, and ask yourself the following questions:

- Will this script only affect the intended files?
- Can unusual or undesirable data be acted upon?
- Can this script be used in unintended ways?
- Can this be used in conjunction with other scripts in a negative manner?
- Will any transactions be adequately logged?

By adequately asking these questions while writing the script, rather than later, you prevent an unfortunate re-write when you need to increase your security. By starting out with this mindset, you won't guarantee the security of your system, but you can help improve it.

You may also want to consider turning off register_globals, magic_quotes, or other convenience settings which may confuse you as to the validity, source, or value of a given variable. Working with PHP in error_reporting(E_ALL) mode can also help warn you about variables being used before they are checked or initialized (so you can prevent unusual data from being operated upon).

General considerations

A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability. If every variable submitted by a user required two forms of biometric validation (such as a

retinal scan and a fingerprint), you would have an extremely high level of accountability. It would also take half an hour to fill out a fairly complex form, which would tend to encourage users to find ways of bypassing the security.

The best security is often inobtrusive enough to suit the requirements without the user being prevented from accomplishing their work, or over-burdening the code author with excessive complexity. Indeed, some security attacks are merely exploits of this kind of overly built security, which tends to erode over time.

A phrase worth remembering: A system is only as good as the weakest link in a chain. If all transactions are heavily logged based on time, location, transaction type, etc. but the user is only verified based on a single cookie, the validity of tying the users to the transaction log is severely weakened.

When testing, keep in mind that you will not be able to test all possibilities for even the simplest of pages. The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard. This is why it’s best to look at the code from a logical perspective, to discern where unexpected data can be introduced, and then follow how it is modified, reduced, or amplified.

The Internet is filled with people trying to make a name for themselves by breaking your code, crashing your site, posting inappropriate content, and otherwise making your day interesting. It doesn’t matter if you have a small or large site, you are a target by simply being online, by having a server that can be connected to. Many cracking programs do not discern by size, they simply trawl massive IP blocks looking for victims. Try not to become one.

Keeping Current

PHP, like any other large system, is under constant scrutiny and improvement. Each new version will often include both major and minor changes to enhance and repair security flaws, configuration mishaps, and other issues that will affect the overall security and stability of your system.

Like other system-level scripting languages and programs, the best approach is to update often, and maintain awareness of the latest versions and their changes.

Část II. Reference jazyka

Kapitola 5. Základní syntaxe

Opuštění HTML

Jsou čtyři způsoby jak opustit HTML a vstoupit to "PHP módu":

Příklad 5-1. Způsoby opuštění HTML

```
1.  <? echo ("toto je nejjednodušší SGML zpracovatelská instrukce\n"); ?>

2.  <?php echo("pokud chcete zpracovávat XHTML nebo XML dokumenty, používe-
jte toto\n"); ?>

3.  <script language="php">
        echo ("některé editory (například FrontPage) nemají zpracovatelské in-
strukce v lásce");
    </script>

4.  <% echo ("případně můžete používat ASP tagy"); %>
    <%= $variable; # toto je zkratka za "<%echo .." %>
```

První způsob je dostupný pouze, pokud jsou povoleny krátké tagy. To se dá udělat povolením konfigurační direktivy short_open_tag v konfiguračním souboru PHP, nebo kompilací PHP s **configure** volbou –enable-short-tags.

Obecně preferovanou metodou je druhý způsob, protože umožňuje snadnou implementaci dalšího generování XHTML z PHP.

Čtvrtý způsob je dostupný, pouze pokud byly povoleny ASP tagy pomocí konfigurační direktivy asp_tags.

Poznámka: Podpora ASP tagů byla přidána v 3.0.4.

Případná bezprostředně následující sekvence konce řádku je součástí uzavírajícího tagu.

Oddělování instrukcí

Instrukce se oddělují stejně jako v C nebo Perlu - ukončujte každý výraz středníkem.

Uzavírající tag (?>) také implikuje konec výrazu, takže následující ukázky jsou ekvivalentní:

```
<?php
    echo "Toto je test";
?>

<?php echo "Toto je test" ?>
```

Komentáře

PHP podporuje komentářové notace jazyků ’C’, ’C++’ a Unixového shellu. Například:

```
<?php
    echo "Toto je test"; // Toto je jednořádkový komentář typu C++
    /* Toto je víceřádkový komentář
       a ještě jeden komentář */
    echo "Toto je další test";
    echo "Poslední Test"; # Toto je komentář shellového typu
?>
```

Jednořádkové typy komentářů ve skutečnosti komentují do konce řádku nebo současného bloku PHP kódu, podle toho, co se vyskytuje dřív.

```
<h1>Toto je  <?php # echo "malý";?> příklad.</h1>
<p>Hlavička na předchozím řádku bude 'Toto je příklad'.
```

Měli byste si dát pozor na vnořování komentářů typu 'C++', ke kterému může dojít při zakomentování velkých bloků.

```
<?php
 /*
    echo "Toto je test"; /* Tento komentář způsobí problém */
 */
?>
```

Kapitola 6. Types

PHP supports the following types:

- array
- floating-point numbers
- integer
- object
- string

The type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used.

If you would like to force a variable to be converted to a certain type, you may either cast the variable or use the **settype()** function on it.

Note that a variable may behave in different manners in certain situations, depending on what type it is at the time. For more information, see the section on Type Juggling.

Integers

Integers can be specified using any of the following syntaxes:

```
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x12; # hexadecimal number (equivalent to 18 decimal)
```

The size of an integer is platform-dependent, although a maximum value of about 2 billion is the usual value (that's 32 bits signed).

Floating point numbers

Floating point numbers ("doubles") can be specified using any of the following syntaxes:

```
$a = 1.234; $a = 1.2e3;
```

The size of a floating point number is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that's 64 bit IEEE format).

> **Varování**
>
> It is quite usual that simple decimal fractions like `0.1` or `0.7` cannot be converted into their internal binary counterparts without a little loss of precision. This can lead to confusing results: for example, `floor((0.1+0.7)*10)` will usually return `7` instead of the expected `8` as the result of the internal representation really being something like `7.9999999999...`.
>
> This is related to the fact that it is impossible to exactly express some fractions in decimal notation with a finite number of digits. For instance, `1/3` in decimal form becomes `0.3333333. . ..`
>
> So never trust floating number results to the last digit and never compare floating point numbers for equality. If you really need higher precision, you should use the arbitrary precision math functions or gmp functions instead.

Strings

Strings can be specified using one of two sets of delimiters.

If the string is enclosed in double-quotes ("), variables within the string will be expanded (subject to some parsing limitations). As in C and Perl, the backslash ("\") character can be used in specifying special characters:

Tabulka 6-1. Escaped characters

sequence	meaning
`\n`	linefeed (LF or 0x0A in ASCII)
`\r`	carriage return (CR or 0x0D in ASCII)
`\t`	horizontal tab (HT or 0x09 in ASCII)
`\\`	backslash
`\$`	dollar sign
`\"`	double-quote
`\[0-7]{1,3}`	the sequence of characters matching the regular expression is a character in octal notation
`\x[0-9A-Fa-f]{1,2}`	the sequence of characters matching the regular expression is a character in hexadecimal notation

If you attempt to escape any other character, both the backslash and the character will be output. In PHP 3, a warning will be issued at the `E_NOTICE` level when this happens. In PHP 4, no warning is generated.

The second way to delimit a string uses the single-quote ("'") character. When a string is enclosed in single quotes, the only escapes that will be understood are "\\" and "\'". This is for convenience, so that you can have single-quotes and backslashes in a single-quoted string. Variables will *not* be expanded inside a single-quoted string.

Another way to delimit strings is by using here doc syntax ("«<"). One should provide an identifier after «<, then the string, and then the same identifier to close the quotation.

The closing identifier *must* begin in the first column of the line. Also, the identifier used must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Here doc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded, but the same care must be taken when expressing complex variables inside a here doc as with strings.

Příklad 6-1. Here doc string quoting example

```
<?php
$str = «<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo {
    var $foo;
    var $bar;

    function foo() {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';

echo «<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Poznámka: Here doc support was added in PHP 4.

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.

Characters within strings may be accessed by treating the string as a numerically-indexed array of characters, using C-like syntax. See below for examples.

Příklad 6-2. Some string examples

```
<?php
/* Assigning a string. */
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';

/* Get the first character of a string  */
$str = 'This is a test.';
$first = $str[0];

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
?>
```

String conversion

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

The string will evaluate as a double if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

When the first expression is a string, the type of the variable will depend on the second expression.

```
$foo = 1 + "10.5";              // $foo is double (11.5)
$foo = 1 + "-1.3e3";            // $foo is double (-1299)
$foo = 1 + "bob-1.3e3";         // $foo is integer (1)
$foo = 1 + "bob3";              // $foo is integer (1)
$foo = 1 + "10 Small Pigs";     // $foo is integer (11)
$foo = 1 + "10 Little Piggies"; // $foo is integer (11)
$foo = "10.0 pigs " + 1;        // $foo is integer (11)
$foo = "10.0 pigs " + 1.0;      // $foo is double (11)
```

For more information on this conversion, see the Unix manual page for strtod(3).

If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

Arrays

Arrays actually act like both hash tables (associative arrays) and indexed arrays (vectors).

Single Dimension Arrays

PHP supports both scalar and associative arrays. In fact, there is no difference between the two. You can create an array using the **list()** or **array()** functions, or you can explicitly set each array element value.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

You can also create an array by simply adding values to the array. When you assign a value to an array variable using empty brackets, the value will be added onto the end of the array.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Arrays may be sorted using the **asort()**, **arsort()**, **ksort()**, **rsort()**, **sort()**, **uasort()**, **usort()**, and **uksort()** functions depending on the type of sort you want.

You can count the number of items in an array using the **count()** function.

You can traverse an array using **next()** and **prev()** functions. Another common way to traverse an array is to use the **each()** function.

Multi-Dimensional Arrays

Multi-dimensional arrays are actually pretty simple. For each dimension of the array, you add another [key] value to the end:

```
$a[1]      = $f;              # one dimensional examples
$a["foo"]  = $f;

$a[1][0]     = $f;            # two dimensional
$a["foo"][2] = $f;            # (you can mix numeric and associative indices)
$a[3]["bar"] = $f;            # (you can mix numeric and associative indices)

$a["foo"][4]["bar"][0] = $f;   # four dimensional!
```

In PHP 3 it is not possible to reference multidimensional arrays directly within strings. For instance, the following will not have the desired result:

```
$a[3]['bar'] = 'Bob';
echo "This won't work: $a[3][bar]";
```

In PHP 3, the above will output `This won't work: Array[bar]`. The string concatenation operator, however, can be used to overcome this:

```
$a[3]['bar'] = 'Bob';
echo "This will work: " . $a[3][bar];
```

In PHP 4, however, the whole problem may be circumvented by enclosing the array reference (inside the string) in curly braces:

```
$a[3]['bar'] = 'Bob';
echo "This will work: {$a[3][bar]}";
```

You can "fill up" multi-dimensional arrays in many ways, but the trickiest one to understand is how to use the **array()** command for associative arrays. These two snippets of code fill up the one-dimensional array in the same way:

```
# Example 1:

$a["color"] = "red";
$a["taste"] = "sweet";
$a["shape"] = "round";
$a["name"] = "apple";
$a[3] = 4;

# Example 2:
$a = array(
     "color" => "red",
     "taste" => "sweet",
     "shape" => "round",
     "name"  => "apple",
     3       => 4
);
```

The **array()** function can be nested for multi-dimensional arrays:

```
<?php
$a = array(
     "apple"  => array(
          "color"  => "red",
          "taste"  => "sweet",
          "shape"  => "round"
     ),
     "orange"  => array(
          "color"  => "orange",
          "taste"  => "tart",
          "shape"  => "round"
     ),
     "banana"  => array(
          "color"  => "yellow",
          "taste"  => "paste-y",
          "shape"  => "banana-shaped"
     )
);

echo $a["apple"]["taste"];    # will output "sweet"
?>
```

Objects

Object Initialization

To initialize an object, you use the `new` statement to instantiate the object to a variable.

```
<?php
class foo {
    function do_foo() {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

For a full discussion, please read the section Classes and Objects.

Type Juggling

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which that variable is used. That is to say, if you assign a string value to variable *`var`*, *`var`* becomes a string. If you then assign an integer value to *`var`*, it becomes an integer.

An example of PHP's automatic type conversion is the addition operator '+'. If any of the operands is a double, then all operands are evaluated as doubles, and the result will be a double. Otherwise, the operands will be interpreted as integers, and the result will also be an integer. Note that this does NOT change the types of the operands themselves; the only change is in how the operands are evaluated.

```
$foo = "0";  // $foo is string (ASCII 48)
$foo++;      // $foo is the string "1" (ASCII 49)
$foo += 1;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a double (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
```

If the last two examples above seem odd, see String conversion.

If you wish to force a variable to be evaluated as a certain type, see the section on Type casting. If you wish to change the type of a variable, see **settype()**.

If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

Poznámka: The behaviour of an automatic conversion to array is currently undefined.

```
$a = 1;       // $a is an integer
$a[0] = "f";  // $a becomes an array, with $a[0] holding "f"
```

While the above example may seem like it should clearly result in $a becoming an array, the first element of which is 'f', consider this:

```
$a = "1";     // $a is a string
$a[0] = "f";  // What about string offsets? What happens?
```

Since PHP supports indexing into strings via offsets using the same syntax as array indexing, the example above leads to a problem: should $a become an array with its first element being "f", or should "f" become the first character of the string $a?

For this reason, as of PHP 3.0.12 and PHP 4.0b3-RC4, the result of this automatic conversion is considered to be undefined. Fixes are, however, being discussed.

Type Casting

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```
$foo = 10;   // $foo is an integer
$bar = (double) $foo;   // $bar is a double
```

The casts allowed are:

- (int), (integer) - cast to integer
- (real), (double), (float) - cast to double
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

Note that tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

It may not be obvious exactly what will happen when casting between certain types. For instance, the following should be noted.

When casting from a scalar or a string variable to an array, the variable will become the first element of the array:

```
$var = ’ciao’;
$arr = (array) $var;
echo $arr[0];  // outputs ’ciao’
```

When casting from a scalar or a string variable to an object, the variable will become an attribute of the object; the attribute name will be ’scalar’:

```
$var = ’ciao’;
$obj = (object) $var;
echo $obj->scalar;  // outputs ’ciao’
```

Kapitola 7. Variables

Basics

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: ’[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*’

> **Poznámka:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = ’not yet’;     // invalid; starts with a number
$_4site = ’not yet’;    // valid; starts with an underscore
$täyte = ’mansikka’;    // valid; ’ä’ is ASCII 228.
```

In PHP 3, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable’s value to another, changing one of those variables will have no effect on the other. For more information on this kind of assignment, see Expressions.

PHP 4 offers another way to assign values to variables: *assign by reference*. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa. This also means that no copying is performed; thus, the assignment happens more quickly. However, any speedup will likely be noticed only in tight loops or when assigning large arrays or objects.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs ’My name is Bob’ twice:

```
<?php
$foo = ’Bob’;              // Assign the value ’Bob’ to $foo
$bar = &$foo;          // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $foo;                 // $foo is altered too.
echo $bar;
?>
```

One important thing to note is that only named variables may be assigned by reference.

```
<?php
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test() {
   return 25;
}

$bar = &test();    // Invalid.
?>
```

Predefined variables

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command-line.

Despite these factors, here is a list of predefined variables available under a stock installation of PHP 3 running as a module under a stock installation of Apache (http://www.apache.org/) 1.3.6.

For a list of all predefined variables (and lots of other useful information), please see (and use) **phpinfo()**.

> **Poznámka:** This list is neither exhaustive nor intended to be. It is simply a guideline as to what sorts of predefined variables you can expect to have access to in your script.

Apache variables

These variables are created by the Apache (http://www.apache.org/) webserver. If you are running another webserver, there is no guarantee that it will provide the same variables; it may omit some, or provide others not listed here. That said, a large number of these variables are accounted for in the CGI 1.1 specification (http://hoohoo.ncsa.uiuc.edu/cgi/env.html), so you should be able to expect those.

Note that few, if any, of these will be available (or indeed have any meaning) if running PHP on the command line.

GATEWAY_INTERFACE

What revision of the CGI specification the server is using; i.e. 'CGI/1.1'.

SERVER_NAME

The name of the server host under which the current script is executing. If the script is running on a virtual host, this will be the value defined for that virtual host.

SERVER_SOFTWARE

Server identification string, given in the headers when responding to requests.

SERVER_PROTOCOL

Name and revision of the information protocol via which the page was requested; i.e. 'HTTP/1.0';

REQUEST_METHOD

Which request method was used to access the page; i.e. 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING

The query string, if any, via which the page was accessed.

DOCUMENT_ROOT

The document root directory under which the current script is executing, as defined in the server's configuration file.

HTTP_ACCEPT

Contents of the `Accept:` header from the current request, if there is one.

HTTP_ACCEPT_CHARSET

Contents of the `Accept-Charset:` header from the current request, if there is one. Example: 'iso-8859-1,*,utf-8'.

HTTP_ACCEPT_ENCODING

Contents of the `Accept-Encoding:` header from the current request, if there is one. Example: 'gzip'.

HTTP_ACCEPT_LANGUAGE

Contents of the `Accept-Language:` header from the current request, if there is one. Example: 'en'.

HTTP_CONNECTION

Contents of the `Connection:` header from the current request, if there is one. Example: 'Keep-Alive'.

HTTP_HOST

Contents of the `Host:` header from the current request, if there is one.

HTTP_REFERER

The address of the page (if any) which referred the browser to the current page. This is set by the user's browser; not all browsers will set this.

HTTP_USER_AGENT

Contents of the `User_Agent:` header from the current request, if there is one. This is a string denoting the browser software being used to view the current page; i.e. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Among other things, you can use this value with **get_browser()** to tailor your page's functionality to the capabilities of the user's browser.

REMOTE_ADDR

The IP address from which the user is viewing the current page.

REMOTE_PORT

The port being used on the user's machine to communicate with the web server.

SCRIPT_FILENAME

The absolute pathname of the currently executing script.

SERVER_ADMIN

The value given to the SERVER_ADMIN (for Apache) directive in the web server configuration file. If the script is running on a virtual host, this will be the value defined for that virtual host.

SERVER_PORT

The port on the server machine being used by the web server for communication. For default setups, this will be '80'; using SSL, for instance, will change this to whatever your defined secure HTTP port is.

SERVER_SIGNATURE

String containing the server version and virtual host name which are added to server-generated pages, if enabled.

PATH_TRANSLATED

Filesystem- (not document root-) based path to the current script, after the server has done any virtual-to-real mapping.

SCRIPT_NAME

Contains the current script's path. This is useful for pages which need to point to themselves.

REQUEST_URI

The URI which was given in order to access this page; for instance, '/index.html'.

Environment variables

These variables are imported into PHP's global namespace from the environment under which the PHP parser is running. Many are provided by the shell under which PHP is running and different systems are likely running different kinds of shells, a definitive list is impossible. Please see your shell's documentation for a list of defined environment variables.

Other environment variables include the CGI variables, placed there regardless of whether PHP is running as a server module or CGI processor.

PHP variables

These variables are created by PHP itself. The `$HTTP_*_VARS` variables are available only if the track_vars configuration is turned on. When enabled, the variables are always set, even if they are empty arrays. This prevents a malicious user from spoofing these variables.

Poznámka: As of PHP 4.0.3, track_vars is always turned on, regardless of the configuration file setting.

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the `$HTTP_*_VARS` arrays. This feature should be used with care, and turned off if possible; while the `$HTTP_*_VARS` variables are safe, the bare global equivalents can be overwritten by user input, with possibly malicious intent. If you cannot turn off register_globals, you must take whatever steps are necessary to ensure that the data you are using is safe.

argv

Array of arguments passed to the script. When the script is run on the command line, this gives C-style access to the command line parameters. When called via the GET method, this will contain the query string.

argc

Contains the number of command line parameters passed to the script (if run on the command line).

PHP_SELF

The filename of the currently executing script, relative to the document root. If PHP is running as a command-line processor, this variable is not available.

HTTP_COOKIE_VARS

An associative array of variables passed to the current script via HTTP cookies.

HTTP_GET_VARS

An associative array of variables passed to the current script via the HTTP GET method.

HTTP_POST_VARS

An associative array of variables passed to the current script via the HTTP POST method.

HTTP_POST_FILES

An associative array of variables containing information about files uploaded via the HTTP POST method. See POST method uploads for information on the contents of `$HTTP_POST_FILES`.

`$HTTP_POST_FILES` is available only in PHP 4.0.0 and later.

HTTP_ENV_VARS

An associative array of variables passed to the current script via the parent environment.

HTTP_SERVER_VARS

An associative array of variables passed to the current script from the HTTP server. These variables are analogous to the Apache variables described above.

Variable scope

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope. This single scope spans included and required files as well. For example:

```
$a = 1;
include "b.inc";
```

Here the `$a` variable will be available within the included `b.inc` script. However, within user-defined functions a local function scope is introduced. Any variable used inside a function is by default limited to the local function scope. For example:

```
$a = 1; /* global scope */
```

```
Function Test () {
    echo $a; /* reference to local scope variable */
}

Test ();
```

This script will not produce any output because the echo statement refers to a local version of the `$a` variable, and it has not been assigned a value within this scope. You may notice that this is a little bit different from the C language in that global variables in C are automatically available to functions unless specifically overridden by a local definition. This can cause some problems in that people may inadvertently change a global variable. In PHP global variables must be declared global inside a function if they are going to be used in that function. An example:

```
$a = 1;
$b = 2;

Function Sum () {
    global $a, $b;

    $b = $a + $b;
}

Sum ();
echo $b;
```

The above script will output "3". By declaring `$a` and `$b` global within the function, all references to either variable will refer to the global version. There is no limit to the number of global variables that can be manipulated by a function.

A second way to access variables from the global scope is to use the special PHP-defined `$GLOBALS` array. The previous example can be rewritten as:

```
$a = 1;
$b = 2;

Function Sum () {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum ();
echo $b;
```

The `$GLOBALS` array is an associative array with the name of the global variable being the key and the contents of that variable being the value of the array element.

Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Consider the following example:

```
Function Test () {
    $a = 0;
    echo $a;
    $a++;
}
```

This function is quite useless since every time it is called it sets `$a` to `0` and prints "0". The `$a++` which increments the variable serves no purpose since as soon as the function exits the `$a` variable disappears. To make a useful counting function which will not lose track of the current count, the `$a` variable is declared static:

```
Function Test () {
    static $a = 0;
    echo $a;
    $a++;
}
```

Now, every time the Test() function is called it will print the value of `$a` and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion. The following simple function recursively counts to 10, using the static variable `$count` to know when to stop:

```
Function Test () {
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count-;
}
```

Variable variables

Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. A normal variable is set with a statement such as:

```
$a = "hello";
```

A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be used as the name of a variable by using two dollar signs. i.e.

```
$$a = "world";
```

At this point two variables have been defined and stored in the PHP symbol tree: `$a` with contents "hello" and `$hello` with contents "world". Therefore, this statement:

```
echo "$a ${$a}";
```

produces the exact same output as:

```
echo "$a $hello";
```

i.e. they both produce: `hello world`.

In order to use variable variables with arrays, you have to resolve an ambiguity problem. That is, if you write `$$a[1]` then the parser needs to know if you meant to use `$a[1]` as a variable, or if you wanted `$$a` as the variable and then the [1] index from that variable. The syntax for resolving this ambiguity is: `${$a[1]}` for the first case and `${$a}[1]` for the second.

Variables from outside PHP

HTML Forms (GET and POST)

When a form is submitted to a PHP script, any variables from that form will be automatically made available to the script by PHP. If the track_vars configuration option is turned on, then these variables will be located in the associative arrays `$HTTP_POST_VARS`, `$HTTP_GET_VARS`, and/or `$HTTP_POST_FILES`, according to the source of the variable in question.

For more information on these variables, please read Predefined variables.

Příklad 7-1. Simple form variable

```
<form action="foo.php" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>
```

When the above form is submitted, the value from the text input will be available in `$HTTP_POST_VARS['username']`. If the register_globals configuration directive is turned on, then the variable will also be available as `$username` in the global scope.

PHP also understands arrays in the context of form variables. You may, for example, group related variables together, or use this feature to retrieve values from a multiple select input:

Příklad 7-2. More complex form variables

```
<form action="array.php" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        <option value="stuttgarter">Stuttgarter Schwabenbräu
        </select>
    <input type="submit">
</form>
```

In PHP 3, the array form variable usage is limited to single-dimensional arrays. In PHP 4, no such restriction applies.

IMAGE SUBMIT variable names

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type=image src="image.gif" name="sub">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables, sub_x and sub_y. These contain the coordinates of the user click within the image. The experienced may note that the actual variable names sent by the browser contains a period rather than an underscore, but PHP converts the period to an underscore automatically.

HTTP Cookies

PHP transparently supports HTTP cookies as defined by Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the **SetCookie()** function. Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. This is the same restriction as for the **Header()** function. Any cookies sent to you from the client will automatically be turned into a PHP variable just like GET and POST method data.

If you wish to assign multiple values to a single cookie, just add *[]* to the cookie name. For example:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Note that a cookie will replace a previous cookie by the same name in your browser unless the path or domain is different. So, for a shopping cart application you may want to keep a counter and pass this along. i.e.

Příklad 7-3. SetCookie Example

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

Environment variables

PHP automatically makes environment variables available as normal PHP variables.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

Since information coming in via GET, POST and Cookie mechanisms also automatically create PHP variables, it is sometimes best to explicitly read a variable from the environment in order to make sure that you are getting the right version. The **getenv()** function can be used for this. You can also set an environment variable with the **putenv()** function.

Dots in incoming variable names

Typically, PHP does not alter the names of variables when they are passed into a script. However, it should be noted that the dot (period, full stop) is not a valid character in a PHP variable name. For the reason, look at it:

```
$varname.ext;  /* invalid variable name */
```

Now, what the parser sees is a variable named `$varname`, followed by the string concatenation operator, followed by the barestring (i.e. unquoted string which doesn't match any known key or reserved words) 'ext'. Obviously, this doesn't have the intended result.

For this reason, it is important to note that PHP will automatically replace any dots in incoming variable names with underscores.

Determining variable types

Because PHP determines the types of variables and converts them (generally) as needed, it is not always obvious what type a given variable is at any one time. PHP includes several functions which find out what type a variable is. They are **gettype()**, **is_long()**, **is_double()**, **is_string()**, **is_array()**, and **is_object()**.

Kapitola 8. Konstanty

PHP definuje několik konstant, a poskytuje mechanismus pro definici dalších za běhu. Konstanty se hodně podobají proměnným s výjimkou dvou skutečností: konstanty se musí definovat pomocí **define()** function, a nemohou později nabývat jiných hodnot.

Předdefinované konstanty (dostupné vždy) jsou:

__FILE__

Název souboru skriptu, který je právě čten. Pokud je použita v souboru, který byl include-ován nebo require-ován, obsahuje název include-ovaného, ne rodičovského souboru.

__LINE__

Číslo řádku ve skriptu, který je právě čten. Pokud je použita v include-ovaném nebo require-ovaném souboru, obsahuje pozici v rámci tohoto souboru.

PHP_VERSION

Textové vyjádření verze běžícího PHP parseru, např. ’3.0.8-dev’.

PHP_OS

Název operačního systému, na kterém PHP parser běží, např. ’Linux’.

TRUE

Pravdivá hodnota.

FALSE

Nepravdivá hodnota.

E_ERROR

Označuje neošetřitelnou chybu jinou než parse error.

E_WARNING

Označuje stav, kdy PHP ví, že je něco špatně, ale bude dál pokračovat. Tyto stavy se dají ošetřit v samotném skriptu. Příkladem by byl neplatný regexp ve funkci **ereg()**.

E_PARSE

Parser se zadávíl neplatnou syntaxí skriptu. Ošetření není možné.

E_NOTICE

Došlo k něčemu co by mohlo být chybou. Provádění skriptu pokračuje. Mezi příklady patří textový index pole neopatřený uvozovkami nebo práce s proměnnou, která ještě nebyla definována.

E_ALL

Všechny E_* konstanty shrnuté do jedné. Při použití s **error_reporting()** způsobí hlášení úplně všech problému zaregistrovaných PHP.

E_* konstanty se typicky používají s funkcí **error_reporting()** nastavení hladiny hlášení chyb. Viz všechny tyto konstanty v Ošetření chyb.

Další konstanty můžete definovat pomocí funkce **define()** function.

Všimněte si, že toto jsou konstanty, ne céčkovská makra; konstanty mohou reprezentovat pouze platná skalární data. Note that these are constants, not C-style macros; only valid scalar data may be represented by a constant.

Příklad 8-1. Definice konstant

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // vytiskne "Hello world."
?>
```

Příklad 8-2. Použití __FILE__ a __LINE__

```
<?php
function report_error($file, $line, $message) {
    echo "Došlo k chybě v souboru $file na řádku $line: $message.";
}

report_error(__FILE__,__LINE__, "Něco je špatně!");
?>
```

Kapitola 9. Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "$a = 5", you're assigning '5' into $a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

After this assignment, you'd expect $a's value to be 5 as well, so if you wrote $b = $a, you'd expect it to behave just as if you wrote $b = 5. In other words, $a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Slightly more complex examples for expressions are functions. For instance, consider the following function:

```
function foo () {
    return 5;
}
```

Assuming you're familiar with the concept of functions (if you're not, take a look at the chapter about functions), you'd assume that typing `$c = foo()` is essentially just like writing `$c = 5`, and you're right. Functions are expressions with the value of their return value. Since foo() returns 5, the value of the expression 'foo()' is 5. Usually functions don't just return a static value but compute something.

Of course, values in PHP don't have to be integers, and very often they aren't. PHP supports three scalar value types: integer values, floating point values and string values (scalar values are values that you can't 'break' into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

So far, users of PHP/FI 2 shouldn't feel any change. However, PHP takes expressions much further, in the same way many other languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we've already dealt with, '$a = 5'. It's easy to see that there are two values involved here, the value of the integer constant '5', and the value of $a which is being updated to 5 as well. But the truth is that there's one additional value involved here, and that's the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that '$a = 5', regardless of what it does, is an expression with the value 5. Thus, writing something like '$b = ($a = 5)' is like writing '$a = 5; $b = 5;' (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write '$b = $a = 5'.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP/FI 2 and many other languages may be familiar with the notation of variable++ and variable–. These are increment and decrement operators. In PHP/FI 2, the statement '$a++' has no value (is not an expression), and thus you can't assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment - pre-increment and post-increment. Both pre-increment and post-increment essentially increment the variable, and the effect on the variable is idential. The difference is with the value of the increment expression. Pre-increment, which is written '++$variable', evaluates to the incremented value (PHP increments the variable before reading its value, thus the name 'pre-increment'). Post-increment, which is written '$variable++' evaluates to the original value of $variable, before it was incremented (PHP increments the variable after reading its value, thus the name 'post-increment').

A very common type of expressions are comparison expressions. These expressions evaluate to either 0 or 1, meaning FALSE or TRUE (respectively). PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). These expressions are most commonly used inside conditional execution, such as `if` statements.

The last example of expressions we'll deal with here is combined operator-assignment expressions. You already know that if you want to increment $a by 1, you can simply write '$a++' or '++$a'. But what if you want to add more than one to it, for instance 3? You could write '$a++' multiple times, but this is obviously not a very efficient or comfortable way. A much more common practice is to write '$a = $a + 3'. '$a + 3' evaluates to the value of $a plus 3, and is assigned back into $a, which results in incrementing $a by 3. In PHP, as in several other languages like C, you can write this in a shorter way, which with time would become clearer and quicker to understand as well. Adding 3 to the current value of $a can be written '$a += 3'. This means exactly "take the value of $a, add 3 to it, and assign it back into $a". In addition to being shorter and clearer, this also results in faster execution. The value of '$a += 3', like the value of a regular assignment, is the assigned value. Notice that it is NOT 3, but the combined value of $a plus 3 (this is the value that's assigned into $a). Any two-place operator can be used in this operator-assignment mode, for example '$a -= 5' (subtract 5 from the value of $a), '$b *= 7' (multiply the value of $b by 7), etc.

There is one more expression that may seem odd if you haven't seen it in other languages, the ternary conditional operator:

```
$first ? $second : $third
```

If the value of the first subexpression is true (non-zero), then it the second subexpression is evaluated, and that is the result of the conditional expression. Otherwise, the third subexpression is evaluated, and that is the value.

The following example should help you understand pre- and post-increment and expressions in general a bit better:

```
function double($i) {
    return $i*2;
}
$b = $a = 5;        /* assign the value five into the variable $a and $b */
$c = $a++;          /* post-increment, assign original value of $a
                       (5) to $c */
$e = $d = ++$b;     /* pre-increment, assign the incremented value of
                       $b (6) to $d and $e */

/* at this point, both $d and $e are equal to 6 */

$f = double($d++);  /* assign twice the value of $d before
                       the increment, 2*6 = 12 to $f */
$g = double(++$e);  /* assign twice the value of $e after
                       the increment, 2*7 = 14 to $g */
$h = $g += 10;      /* first, $g is incremented by 10 and ends with the
                       value of 24. the value of the assignment (24) is
                       then assigned into $h, and $h ends with the value
                       of 24 as well. */
```

In the beginning of the chapter we said that we'll be describing the various statement types, and as promised, expressions can be statements. However, not every expression is a statement. In this case, a statement has the form of 'expr' ';' that is, an expression followed by a semicolon. In '$b=$a=5;', $a=5 is a valid expression, but it's not a statement by itself. '$b=$a=5;' however is a valid statement.

One last thing worth mentioning is the truth value of expressions. In many events, mainly in conditional execution and loops, you're not interested in the specific value of the expression, but only care about whether it means TRUE or FALSE (PHP doesn't have a dedicated boolean type). The truth value of expressions in PHP is calculated in a similar way to perl. Any numeric non-zero numeric value is TRUE, zero is FALSE. Be sure to note that negative values are non-zero and are thus considered TRUE! The empty string and the string "0" are FALSE; all other strings are TRUE. With non-scalar values (arrays and objects) - if the value contains no elements it's considered FALSE, otherwise it's considered TRUE.

PHP provides a full and powerful implementation of expressions, and documenting it entirely goes beyond the scope of this manual. The above examples should give you a good idea about what expressions are and how you can construct useful expressions. Throughout the rest of this manual we'll write *`expr`* to indicate any valid PHP expression.

Kapitola 10. Operators

Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Tabulka 10-1. Arithmetic Operators

Example	Name	Result
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulus	Remainder of $a divided by $b.

The division operator ("/") returns an integer value (the result of an integer division) if the two operands are integers (or strings that get converted to integers) and the quotient is an integer. If either operand is a floating-point value, or the operation results in a non-integer value, a floating-point value is returned.

Assignment Operators

The basic assignment operator is "=". Your first inclination might be to think of this as "equal to". Don't. It really means that the the left operand gets set to the value of the expression on the rights (that is, "gets set to").

The value of an assignment expression is the value assigned. That is, the value of "$a = 3" is 3. This allows you to do some tricky things:

```
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
```

In addition to the basic assignment operator, there are "combined operators" for all of the binary arithmetic and string operators that allow you to use a value in an expression and then set its value to the result of that expression. For example:

```
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // sets $b to "Hello There!", just like $b = $b . "There!";
```

Note that the assignment copies the original variable to the new one (assignment by value), so changes to one will not affect the other. This may also have relevance if you need to copy something like a large array inside a tight loop. PHP 4 supports assignment by reference, using the `$var = &$othervar;` syntax, but this is not possible in PHP 3. 'Assignment by reference' means that both variables end up pointing at the same data, and nothing is copied anywhere. To learn more about references, please read References explained.

Bitwise Operators

Bitwise operators allow you to turn specific bits within an integer on or off.

Tabulka 10-2. Bitwise Operators

Example	Name	Result
$a & $b	And	Bits that are set in both $a and $b are set.
$a | $b	Or	Bits that are set in either $a or $b are set.

Example	Name	Result
$a ^ $b	Xor	Bits that are set in $a or $b but not both are set.
~ $a	Not	Bits that are set in $a are not set, and vice versa.
$a « $b	Shift left	Shift the bits of $a $b steps to the left (each step means "multiply by two")
$a » $b	Shift right	Shift the bits of $a $b steps to the right (each step means "divide by two")

Comparison Operators

Comparison operators, as their name implies, allow you to compare two values.

Tabulka 10-3. Comparison Operators

Example	Name	Result
$a == $b	Equal	True if $a is equal to $b.
$a === $b	Identical	True if $a is equal to $b, and they are of the same type. (PHP 4 only)
$a != $b	Not equal	True if $a is not equal to $b.
$a !== $b	Not identical	True if $a is not equal to $b, or they are not of the same type. (PHP 4 only)
$a < $b	Less than	True if $a is strictly less than $b.
$a > $b	Greater than	True if $a is strictly greater than $b.
$a <= $b	Less than or equal to	True if $a is less than or equal to $b.
$a >= $b	Greater than or equal to	True if $a is greater than or equal to $b.

Another conditional operator is the "?:" (or ternary) operator, which operates as in C and many other languages.

```
(expr1) ? (expr2) : (expr3);
```

This expression evaluates to *`expr2`* if *`expr1`* evaluates to true, and *`expr3`* if *`expr1`* evaluates to false.

Error Control Operators

PHP supports one error control operator: the at sign (@). When prepended to an expression in PHP, any error messages that might be generated by that expression will be ignored.

If the track_errors feature is enabled, any error message generated by the expression will be saved in the global variable $php_errormsg. This variable will be overwritten on each error, so check early if you want to use it.

```
<?php
/* Intentional SQL error (extra quote): */
$res = @mysql_query ("select name, code from 'namelist") or
    die ("Query failed: error was '$php_errormsg'");
?>
```

See also **error_reporting()**.

> **Varování**
>
> Currently the "@" error-control operator prefix will even disable error reporting for critical errors that will terminate script execution. Among other things, this means that if you use "@" to suppress errors from a certain function and either it isn't available or has been mistyped, the script will die right there with no indication as to why.

Execution Operators

PHP supports one execution operator: backticks (“). Note that these are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command; the output will be returned (i.e., it won't simply be dumped to output; it can be assigned to a variable).

```
$output = ‘ls -al‘;
echo "<pre>$output</pre>";
```

Poznámka: The backtick operator is disabled when safe mode is enabled.

See also **system()**, **passthru()**, **exec()**, **popen()**, and **escapeshellcmd()**.

Incrementing/Decrementing Operators

PHP supports C-style pre- and post-increment and decrement operators.

Tabulka 10-4. Increment/decrement Operators

Example	Name	Effect
++$a	Pre-increment	Increments $a by one, then returns $a.
$a++	Post-increment	Returns $a, then increments $a by one.
–$a	Pre-decrement	Decrements $a by one, then returns $a.
$a–	Post-decrement	Returns $a, then decrements $a by one.

Here's a simple example script:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++ . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a- . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";

echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . -$a . "<br>\n";
```

```
echo "Should be 4: " . $a . "<br>\n";
?>
```

Logical Operators

Tabulka 10-5. Logical Operators

Example	Name	Result
$a and $b	And	True if both $a and $b are true.
$a or $b	Or	True if either $a or $b is true.
$a xor $b	Xor	True if either $a or $b is true, but not both.
! $a	Not	True if $a is not true.
$a && $b	And	True if both $a and $b are true.
$a || $b	Or	True if either $a or $b is true.

The reason for the two different variations of "and" and "or" operators is that they operate at different precedences. (See Operator Precedence.)

Operator Precedence

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression `1 + 5 * 3`, the answer is `16` and not `18` because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: `(1 + 5) * 3` evaluates to `18`.

The following table lists the precedence of operators with the lowest-precedence operators listed first.

Tabulka 10-6. Operator Precedence

Associativity	Operators
left	,
left	or
left	xor
left	and
right	print
left	= += -= *= /= .= %= &= |= ^= ~= «= »=
left	? :
left	||
left	&&
left	|
left	^
left	&
non-associative	== != === !==
non-associative	< <= > >=
left	« »
left	+ - .
left	* / %
right	! ~ ++ – (int) (double) (string) (array) (object) @

Associativity	Operators
right	[
non-associative	new

String Operators

There are two string operators. The first is the concatenation operator (’.’), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator (’.=’), which appends the argument on the right side to the argument on the left side. Please read Assignment Operators for more information.

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
```

Kapitola 11. Control Structures

Any PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement of even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well. The various statement types are described in this chapter.

if

The `if` construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an `if` structure that is similar to that of C:

```
if (expr)
    statement
```

As described in the section about expressions, expr is evaluated to its truth value. If *`expr`* evaluates to `TRUE`, PHP will execute statement, and if it evaluates to `FALSE` - it'll ignore it.

The following example would display `a is bigger than b` if *`$a`* is bigger than *`$b`*:

```
if ($a > $b)
    print "a is bigger than b";
```

Often you'd want to have more than one statement to be executed conditionally. Of course, there's no need to wrap each statement with an `if` clause. Instead, you can group several statements into a statement group. For example, this code would display `a is bigger than b` if *`$a`* is bigger than *`$b`*, and would then assign the value of *`$a`* into *`$b`*:

```
if ($a > $b) {
    print "a is bigger than b";
    $b = $a;
}
```

If statements can be nested indefinitely within other `if` statements, which provides you with complete flexibility for conditional execution of the various parts of your program.

else

Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what `else` is for. `else` extends an `if` statement to execute a statement in case the expression in the `if` statement evaluates to `FALSE`. For example, the following code would display `a is bigger than b` if *`$a`* is bigger than *`$b`*, and `a is NOT bigger than b` otherwise:

```
if ($a > $b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
```

The `else` statement is only executed if the `if` expression evaluated to `FALSE`, and if there were any `elseif` expressions - only if they evaluated to `FALSE` as well (see elseif).

elseif

`elseif`, as its name suggests, is a combination of `if` and `else`. Like `else`, it extends an `if` statement to execute a different statement in case the original `if` expression evaluates to `FALSE`. However, unlike `else`, it will execute that

alternative expression only if the `elseif` conditional expression evaluates to `TRUE`. For example, the following code would display `a is bigger than b`, `a equal to b` or `a is smaller than b`:

```
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
```

There may be several `elseif`s within the same `if` statement. The first `elseif` expression (if any) that evaluates to `true` would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word). The syntactic meaning is slightly different (if you're familiar with C, this is the same behavior) but the bottom line is that both would result in exactly the same behavior.

The `elseif` statement is only executed if the preceding `if` expression and any preceding `elseif` expressions evaluated to `FALSE`, and the current `elseif` expression evaluated to `TRUE`.

Alternative syntax for control structures

PHP offers an alternative syntax for some of its control structures; namely, `if`, `while`, `for`, `foreach`, and `switch`. In each case, the basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to `endif;`, `endwhile;`, `endfor;`, `endforeach;`, or `endswitch;`, respectively.

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

In the above example, the HTML block "A = 5" is nested within an `if` statement written in the alternative syntax. The HTML block would be displayed only if $a is equal to 5.

The alternative syntax applies to `else` and `elseif` as well. The following is an `if` structure with `elseif` and `else` in the alternative format:

```
if ($a == 5):
    print "a equals 5";
    print "...";
elseif ($a == 6):
    print "a equals 6";
    print "!!!";
else:
    print "a is neither 5 nor 6";
endif;
```

See also while, for, and if for further examples.

`while`

`while` loops are the simplest type of loop in PHP. They behave just like their C counterparts. The basic form of a `while` statement is:

```
while (expr) statement
```

The meaning of a `while` statement is simple. It tells PHP to execute the nested statement(s) repeatedly, as long as the `while` expression evaluates to `TRUE`. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the `while` expression evaluates to `FALSE` from the very beginning, the nested statement(s) won't even be run once.

Like with the `if` statement, you can group multiple statements within the same `while` loop by surrounding a group of statements with curly braces, or by using the alternate syntax:

```
while (expr): statement ... endwhile;
```

The following examples are identical, and both print numbers from 1 to 10:

```
/* example 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* the printed value would be
                    $i before the increment
                    (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

`do..while` loops are very similar to `while` loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular `while` loops is that the first iteration of a `do..while` loop is guarenteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular `while` loop (the truth expression is checked at the beginning of each iteration, if it evaluates to `FALSE` right from the beginning, the loop execution would end immediately).

There is just one syntax for `do..while` loops:

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to `FALSE` ($i is not bigger than 0) and the loop execution ends.

Advanced C users may be familiar with a different usage of the `do..while` loop, to allow stopping execution in the middle of code blocks, by encapsulating them with `do..while`(0), and using the `break` statement. The following code fragment demonstrates this:

```
do {
    if ($i < 5) {
        print "i is not big enough";
        break;
    }
    $i *= $factor;
```

```
    if ($i < $minimum_limit) {
        break;
    }
    print "i is ok";

     ...process i...

} while(0);
```

Don’t worry if you don’t understand this right away or at all. You can code scripts and even powerful scripts without using this ‘feature’.

for

`for` loops are the most complex loops in PHP. They behave like their C counterparts. The syntax of a `for` loop is:

```
for (expr1; expr2; expr3) statement
```

The first expression (*`expr1`*) is evaluated (executed) once unconditionally at the beginning of the loop.

In the beginning of each iteration, *`expr2`* is evaluated. If it evaluates to `TRUE`, the loop continues and the nested statement(s) are executed. If it evaluates to `FALSE`, the execution of the loop ends.

At the end of each iteration, *`expr3`* is evaluated (executed).

Each of the expressions can be empty. *`expr2`* being empty means the loop should be run indefinitely (PHP implicitly considers it as `TRUE`, like C). This may not be as useless as you might think, since often you’d want to end the loop using a conditional `break` statement instead of using the `for` truth expression.

Consider the following examples. All of them display numbers from 1 to 10:

```
/* example 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* example 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* example 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* example 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Of course, the first example appears to be the nicest one (or perhaps the fourth), but you may find that being able to use empty expressions in `for` loops comes in handy in many occasions.

PHP also supports the alternate "colon syntax" for `for` loops.

```
for (expr1; expr2; expr3): statement; ...; endfor;
```

Other languages have a `foreach` statement to traverse an array or hash. PHP 3 has no such construct; PHP 4 does (see foreach). In PHP 3, you can combine while with the **list()** and **each()** functions to achieve the same effect. See the documentation for these functions for an example.

`foreach`

PHP 4 (not PHP 3) includes a `foreach` construct, much like perl and some other languages. This simply gives an easy way to iterate over arrays. There are two syntaxes; the second is a minor but useful extension of the first:

```
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement
```

The first form loops over the array given by `array_expression`. On each loop, the value of the current element is assigned to `$value` and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).

The second form does the same thing, except that the current element's key will be assigned to the variable `$key` on each loop.

> **Poznámka:** When `foreach` first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call **reset()** before a `foreach` loop.

> **Poznámka:** Also note that `foreach` operates on a copy of the specified array, not the array itself, therefore the array pointer is not modified as with the **each()** construct and changes to the array element returned are not reflected in the original array.

> **Poznámka:** `foreach` does not support the ability to suppress error messages using '@'.

You may have noticed that the following are functionally identical:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
```

The following are also functionally identical:

```
reset ($arr);
```

```
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```

Some more examples to demonstrate usages:

```
/* foreach example 1: value only */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
   print "Current value of \$a: $v.\n";
}

/* foreach example 2: value (with key printed for illustration) */

$a = array (1, 2, 3, 17);

$i = 0; /* for illustrative purposes only */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
}

/* foreach example 3: key and value */

$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach example 4: multi-dimensional arrays */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach example 5: dynamic arrays

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
```

break

`break` ends execution of the current `for`, `while`, `foreach` or `switch` structure.

`break` accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
$arr = array (’one’, ’two’, ’three’, ’four’, ’stop’, ’five’);
while (list (, $val) = each ($arr)) {
    if ($val == ’stop’) {
        break;    /* You could also write ’break 1;’ here. */
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break 1;  /* Exit only the switch. */
    case 10:
        echo "At 10; quitting<br>\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
}
```

continue

`continue` is used within looping structures to skip the rest of the current loop iteration and continue execution at the beginning of the next iteration.

`continue` accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

```
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // skip odd members
        continue;
    }
    do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "  Middle<br>\n";
        while (1) {
            echo "  Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}
```

switch

The switch statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the switch statement is for.

The following two examples are two different ways to write the same thing, one using a series of if statements, and the other using the switch statement:

```
if ($i == 0) {
    print "i equals 0";
}
if ($i == 1) {
    print "i equals 1";
}
if ($i == 2) {
    print "i equals 2";
}

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
```

It is important to understand how the switch statement is executed in order to avoid mistakes. The switch statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a case statement is found with a value that matches the value of the switch expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the switch block, or the first time it sees a break statement. If you don't write a break statement at the end of a case's statement list, PHP will go on executing the statements of the following case. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
    case 1:
        print "i equals 1";
    case 2:
        print "i equals 2";
}
```

Here, if $i equals to 0, PHP would execute all of the print statements! If $i equals to 1, PHP would execute the last two print statements, and only if $i equals to 2, you'd get the 'expected' behavior and only 'i equals 2' would be displayed. So, it's important not to forget break statements (even though you may want to avoid supplying them on purpose under certain circumstances).

In a switch statement, the condition is evaluated only once and the result is compared to each case statement. In an elseif statement, the condition is evaluated again. If your condition is more complicated than a simple compare and/or is in a tight loop, a switch may be faster.

The statement list for a case can also be empty, which simply passes control into the statement list for the next case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
```

```
        print "i is less than 3 but not negative";
        break;
    case 3:
        print "i is 3";
}
```

A special case is the default case. This case matches anything that wasn't matched by the other cases, and should be the last `case` statement. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

The `case` expression may be any expression that evaluates to a simple type, that is, integer or floating-point numbers and strings. Arrays or objects cannot be used here unless they are dereferenced to a simple type.

The alternative syntax for control structures is supported with switches. For more information, see Alternative syntax for control structures .

```
switch ($i):
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
endswitch;
```

require()

The **require()** statement replaces itself with the specified file, much like the C preprocessor's `#include` works.

If "URL fopen wrappers" are enabled in PHP (which they are in the default configuration), you can specify the file to be **require()**ed using an URL instead of a local pathname. See Remote files and **fopen()** for more information.

An important note about how this works is that when a file is **include()**ed or **require()**ed, parsing drops out of PHP mode and into HTML mode at the beginning of the target file, and resumes PHP mode again at the end. For this reason, any code inside the target file which should be executed as PHP code must be enclosed within valid PHP start and end tags.

require() is not actually a function in PHP; rather, it is a language construct. It is subject to some different rules than functions are. For instance, **require()** is not subject to any containing control structures. For another, it does not return any value; attempting to read a return value from a **require()** call results in a parse error.

Unlike **include()**, **require()** will *always* read in the target file, *even if the line it's on never executes*. If you want to conditionally include a file, use **include()**. The conditional statement won't affect the **require()**. However, if the line on which the **require()** occurs is not executed, neither will any of the code in the target file be executed.

Similarly, looping structures do not affect the behaviour of **require()**. Although the code contained in the target file is still subject to the loop, the **require()** itself happens only once.

This means that you can't put a **require()** statement inside of a loop structure and expect it to include the contents of a different file on each iteration. To do that, use an **include()** statement.

```
require ('header.inc');
```

When a file is **require()**ed, the code it contains inherits the variable scope of the line on which the **require()** occurs. Any variables available at that line in the calling file will be available within the called file. If the **require()** occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the **require()**ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the **require()**ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as **require()**ing the file and having it inherit the parent file's variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, 'works' here means that the variables
 * $varone and $vartwo are available within the require()ed file. */

/* Won't work; file.txt wasn't handled by someserver. */
require ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won't work; looks for a file named 'file.php?varone=1&vartwo=2'
 * on the local filesystem. */
require ("file.php?varone=1&vartwo=2");

/* Works. */
require ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
require ("file.txt");  /* Works. */
require ("file.php");  /* Works. */
```

In PHP 3, it is possible to execute a `return` statement inside a **require()**ed file, as long as that statement occurs in the global scope of the **require()**ed file. It may not occur within any block (meaning inside braces ({}). In PHP 4, however, this ability has been discontinued. If you need this functionality, see **include()**.

See also **include()**, **require_once()**, **include_once()**, **readfile()**, and **virtual()**.

include()

The **include()** statement includes and evaluates the specified file.

If "URL fopen wrappers" are enabled in PHP (which they are in the default configuration), you can specify the file to be **include()**ed using an URL instead of a local pathname. See Remote files and **fopen()** for more information.

An important note about how this works is that when a file is **include()**ed or **require()**ed, parsing drops out of PHP mode and into HTML mode at the beginning of the target file, and resumes again at the end. For this reason, any code inside the target file which should be executed as PHP code must be enclosed within valid PHP start and end tags.

This happens each time the **include()** statement is encountered, so you can use an **include()** statement within a looping structure to include a number of different files.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
```

```
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
```

include() differs from **require()** in that the include statement is re-evaluated each time it is encountered (and only when it is being executed), whereas the **require()** statement is replaced by the required file when it is first encountered, whether the contents of the file will be evaluated or not (for example, if it is inside an if statement whose condition evaluated to false).

Because **include()** is a special language construct, you must enclose it within a statement block if it is inside a conditional block.

```
/* This is WRONG and will not work as desired. */

if ($condition)
    include($file);
else
    include($other);

/* This is CORRECT. */

if ($condition) {
    include($file);
} else {
    include($other);
}
```

In both PHP 3 and PHP 4, it is possible to execute a `return` statement inside an **include()**ed file, in order to terminate processing in that file and return to the script which called it. Some differences in the way this works exist, however. The first is that in PHP 3, the `return` may not appear inside a block unless it's a function block, in which case the `return` applies to that function and not the whole file. In PHP 4, however, this restriction does not exist. Also, PHP 4 allows you to return values from **include()**ed files. You can take the value of the **include()** call as you would a normal function. This generates a parse error in PHP 3.

Příklad 11-1. include() in PHP 3 and PHP 4

Assume the existence of the following file (named `test.inc`) in the same directory as the main file:

```
<?php
echo "Before the return <br>\n";
if (1) {
    return 27;
}
echo "After the return <br>\n";
?>
```

Assume that the main file (`main.html`) contains the following:

```
<?php
$retval = include ('test.inc');
echo "File returned: '$retval'<br>\n";
?>
```

When `main.html` is called in PHP 3, it will generate a parse error on line 2; you can't take the value of an **include()** in PHP 3. In PHP 4, however, the result will be:

```
Before the return
File returned: '27'
```

Now, assume that `main.html` has been altered to contain the following:

```
<?php
include (’test.inc’);
echo "Back in main.html<br>\n";
?>
```

In PHP 4, the output will be:

```
Before the return
Back in main.html
```

However, PHP 3 will give the following output:

```
Before the return
27Back in main.html

Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5
```

The above parse error is a result of the fact that the `return` statement is enclosed in a non-function block within `test.inc`. When the return is moved outside of the block, the output is:

```
Before the return
27Back in main.html
```

The spurious ’27’ is due to the fact that PHP 3 does not support `return`ing values from files like that.

When a file is **include()**ed, the code it contains inherits the variable scope of the line on which the **include()** occurs. Any variables available at that line in the calling file will be available within the called file. If the **include()** occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the **include()**ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the **include()**ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as **include()**ing the file and having it inherit the parent file’s variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, ’works’ here means that the variables
 * $varone and $vartwo are available within the include()ed file. */

/* Won’t work; file.txt wasn’t handled by someserver. */
include ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won’t work; looks for a file named ’file.php?varone=1&vartwo=2’
 * on the local filesystem. */
include ("file.php?varone=1&vartwo=2");

/* Works. */
include ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
include ("file.txt");  /* Works. */
include ("file.php");  /* Works. */
```

See also **require()**, **require_once()**, **include_once()**, **readfile()**, and **virtual()**.

require_once()

The **require_once()** statement replaces itself with the specified file, much like the C preprocessor’s `#include` works, and in that respect is similar to the **require()** statement. The main difference is that in an inclusion chain, the use of

require_once() will assure that the code is added to your script only once, and avoid clashes with variable values or function names that can happen.

For example, if you create the following 2 include files `utils.inc` and `foolib.inc`

Příklad 11-2. utils.inc

```
<?php
define(PHPVERSION, floor(phpversion()));
echo "GLOBALS ARE NICE\n";
function goodTea() {
return "Oolong tea tastes good!";
}
?>
```

Příklad 11-3. foolib.inc

```
<?php
require ("utils.inc");
function showVar($var) {
if (PHPVERSION == 4) {
print_r($var);
} else {
var_dump($var);
}
}

// bunch of other functions ...
?>
```

And then you write a script `cause_error_require.php`

Příklad 11-4. cause_error_require.php

```
<?php
require("foolib.inc");
/* the following will generate an error */
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "this is requiring utils.inc again which is also\n";
echo "required in foolib.inc\n";
echo "Running goodTea: ".goodTea()."\n";
echo "Printing foo: \n";
showVar($foo);
?>
```

When you try running the latter one, the resulting ouptut will be (using PHP 4.01pl2):

```
GLOBALS ARE NICE
GLOBALS ARE NICE

Fatal error:  Cannot redeclare goodTea() in utils.inc on line 5
```

By modifying `foolib.inc` and `cause_errror_require.php` to use **require_once()** instead of **require()** and renaming the last one to `avoid_error_require_once.php`, we have:

Příklad 11-5. foolib.inc (fixed)

```
...
require_once("utils.inc");
function showVar($var) {
...
```

Příklad 11-6. avoid_error_require_once.php

```
...
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complex","quaternion"));
...
```

And when running the latter, the output will be (using PHP 4.0.1pl2):

```
GLOBALS ARE NICE
this is requiring globals.inc again which is also
required in foolib.inc
Running goodTea: Oolong tea tastes good!
Printing foo:
Array
(
    [0] => 1
    [1] => Array
        (
            [0] => complex
            [1] => quaternion
        )

)
```

Also note that, analogous to the behavior of the `#include` of the C preprocessor, this statement acts at "compile time", e.g. when the script is parsed and before it is executed, and should not be used for parts of the script that need to be inserted dynamically during its execution. You should use **include_once()** or **include()** for that purpose.

For more examples on using **require_once()** and **include_once()**, look at the PEAR code included in the latest PHP source code distributions.

See also: **require()**, **include()**, **include_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, and **virtual()**.

include_once()

The **include_once()** statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the **include()** statement, with the important difference that if the code from a file has already been included, it will not be included again.

As mentioned in the **require_once()** description, the **include_once()** should be used in the cases in which the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

For more examples on using **require_once()** and **include_once()**, look at the PEAR code included in the latest PHP source code distributions.

include_once() was added in PHP 4.0.1pl2

See also: **require()**, **include()**, **require_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, and **virtual()**.

Kapitola 12. Functions

User-defined functions

A function may be defined using syntax such as the following:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Example function.\n";
    return $retval;
}
```

Any valid PHP code may appear inside a function, even other functions and class definitions.

In PHP 3, functions must be defined before they are referenced. No such requirement exists in PHP 4.

PHP does not support function overloading, nor is it possible to undefine or redefine previously-declared functions.

PHP 3 does not support variable numbers of arguments to functions, although default arguments are supported (see Default argument values for more information). PHP 4 supports both: see Variable-length argument lists and the function references for **func_num_args()**, **func_get_arg()**, and **func_get_args()** for more information.

Function arguments

Information may be passed to functions via the argument list, which is a comma-delimited list of variables and/or constants.

PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are supported only in PHP 4 and later; see Variable-length argument lists and the function references for **func_num_args()**, **func_get_arg()**, and **func_get_args()** for more information. A similar effect can be achieved in PHP 3 by passing an array of arguments to a function:

```
function takes_array($input) {
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Making arguments be passed by reference

By default, function arguments are passed by value (so that if you change the value of the argument within the function, it does not get changed outside of the function). If you wish to allow a function to modify its arguments, you must pass them by reference.

If you want an argument to a function to always be passed by reference, you can prepend an ampersand (&) to the argument name in the function definition:

```
function add_some_extra(&$string) {
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
```

If you wish to pass a variable by reference to a function which does not do this by default, you may prepend an ampersand to the argument name in the function call:

```
function foo ($bar) {
    $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;    // outputs 'This is a string, '
```

```
foo (&$str);
echo $str;    // outputs 'This is a string, and something extra.'
```

Default argument values

A function may define C++-style default values for scalar arguments as follows:

```
function makecoffee ($type = "cappucino") {
    return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

The output from the above snippet is:

```
Making a cup of cappucino.
Making a cup of espresso.
```

The default value must be a constant expression, not (for example) a variable or class member.

Note that when using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected. Consider the following code snippet:

```
function makeyogurt ($type = "acidophilus", $flavour) {
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // won't work as expected
```

The output of the above example is:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

Now, compare the above with this:

```
function makeyogurt ($flavour, $type = "acidophilus") {
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // works as expected
```

The output of this example is:

```
Making a bowl of acidophilus raspberry.
```

Variable-length argument lists

PHP 4 has support for variable-length argument lists in user-defined functions. This is really quite easy, using the **func_num_args()**, **func_get_arg()**, and **func_get_args()** functions.

No special syntax is required, and argument lists may still be explicitly provided with function definitions and will behave as normal.

Returning values

Values are returned by using the optional return statement. Any type may be returned, including lists and objects.

```
function square ($num) {
    return $num * $num;
}
echo square (4);   // outputs '16'.
```

You can't return multiple values from a function, but similar results can be obtained by returning a list.

```
function small_numbers() {
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

To return a reference from a function, you have to use the reference operator & in both the function declaration and when assigning the returned value to a variable:

```
function &returns_reference() {
    return $someref;
}

$newref =&returns_reference();
```

`old_function`

The `old_function` statement allows you to declare a function using a syntax identical to PHP/FI2 (except you must replace 'function' with 'old_function'.

This is a deprecated feature, and should only be used by the PHP/FI2->PHP 3 convertor.

> **Varování**
>
> Functions declared as `old_function` cannot be called from PHP's internal code. Among other things, this means you can't use them in functions such as **usort()**, **array_walk()**, and **register_shutdown_function()**. You can get around this limitation by writing a wrapper function (in normal PHP 3 form) to call the `old_function`.

Variable functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Příklad 12-1. Variable function example

```
<?php
function foo() {
    echo "In foo()<br>\n";
}

function bar( $arg = ” ) {
    echo "In bar(); argument was ’$arg’.<br>\n";
}

$func = ’foo’;
$func();
$func = ’bar’;
$func( ’test’ );
?>
```

Kapitola 13. Classes and Objects

class

A class is a collection of variables and functions working with these variables. A class is defined using the following syntax:

```
<?php
class Cart {
    var $items;  // Items in our shopping cart

    // Add $num articles of $artnr to the cart

    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Take $num articles of $artnr out of the cart

    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

This defines a class named Cart that consists of an associative array of articles in the cart and two functions to add and remove items from this cart.

> **Poznámka:** In PHP 4, only constant initializers for `var` variables are allowed. Use constructors for non-constant initializers.
>
> ```
> /* None of these will work in PHP 4. */
> class Cart {
>     var $todays_date = date("Y-m-d");
>     var $name = $firstname;
>     var $owner = 'Fred ' . 'Jones';
> }
>
> /* This is how it should be done. */
> class Cart {
>     var $todays_date;
>     var $name;
>     var $owner;
>
>     function Cart() {
>         $this->todays_date = date("Y-m-d");
>         $this->name = $GLOBALS['firstname'];
>         /* etc. . . */
>     }
> }
> ```

Classes are types, that is, they are blueprints for actual variables. You have to create a variable of the desired type with the new operator.

```
$cart = new Cart;
$cart->add_item("10", 1);
```

This creates an object $cart of the class Cart. The function add_item() of that object is being called to add 1 item of article number 10 to the cart.

Classes can be extensions of other classes. The extended or derived class has all variables and functions of the base class and what you add in the extended definition. This is done using the extends keyword. Multiple inheritance is not supported.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

This defines a class Named_Cart that has all variables and functions of Cart plus an additional variable $owner and an additional function set_owner(). You create a named cart the usual way and can now set and get the carts owner. You can still use normal cart functions on named carts:

```
$ncart = new Named_Cart;    // Create a named cart
$ncart->set_owner ("kris"); // Name that cart
print $ncart->owner;        // print the cart owners name
$ncart->add_item ("10", 1); // (inherited functionality from cart)
```

Within functions of a class the variable $this means this object. You have to use $this->something to access any variable or function named something within your current object. Both in and outside of the object you do not need a $ when accessing an object's properties.

```
$ncart->owner  = "chris"; // no $

$ncart->$owner = "chris";
// this is invalid because $ncart->$owner = $ncart->""

$myvar = 'owner';
$ncart->$myvar = "chris";
// this is valid because $ncart->$myvar = $ncart->owner
```

Constructors are functions in a class that are automatically called when you create a new instance of a class. A function becomes a constructor when it has the same name as the class.

```
class Auto_Cart extends Cart {
    function Auto_Cart () {
        $this->add_item ("10", 1);
    }
}
```

This defines a class Auto_Cart that is a Cart plus a constructor which initializes the cart with one item of article number "10" each time a new Auto_Cart is being made with "new". Constructors can also take arguments and these arguments can be optional, which makes them much more useful.

```
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}

// Shop the same old boring stuff.

$default_cart   = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart ("20", 17);
```

> **Výstraha**
>
> For derived classes, the constructor of the parent class is not automatically called when the derived class's constructor is called.

References inside the constructor

Creating references within the constructor can lead to confusing results. This tutorial-like section helps you to avoid problems.

```
class foo {
    function foo($name) {
        // create a reference inside the global array $globalref
        global $globalref;
$globalref[] = &$this;
        // set name to passed value
        $this->setName($name);
// and put it out
        $this->echoName();
    }

    function echoName() {
        echo "<br>",$this->Name;
    }

function setName($name) {
$this->Name = $name;
    }
}
```

Let us check out if there is a difference between `$bar1` which has been created using the copy = operator and `$bar2` which has been created using the reference =& operator...

```
    $bar1 = new foo('set in constructor');
    $bar1->echoName();
    $globalref[0]->echoName();

    /* output:
    set in constructor
    set in constructor
    set in constructor */

    $bar2 =& new foo('set in constructor');
    $bar2->echoName();
    $globalref[1]->echoName();

    /* output:
    set in constructor
    set in constructor
    set in constructor */
```

Apparently there is no difference, but in fact there is a very significant one: `$bar1` and `$globalref[0]` are _NOT_ referenced, they are NOT the same variable. This is because "new" does not return a reference by default, instead it returns a copy.

Poznámka: There is no performance loss (since php 4 and up use reference counting) returning copies instead of references. On the contrary it is most often better to simply work with copies instead of references, because creating references takes some time where creating copies virtually takes no time (unless none of them is a large array or object and one of them gets changed and the other(s) one(s) subsequently, then it would be wise to use references to change them all concurrently).

To prove what is written above let us watch the code below.

```
        // now we will change the name. what do you expect?
        // you could expect that both $bar and $globalref[0] change their names...
        $bar1->setName('set from outside');

        // as mentioned before this is not the case.
        $bar1->echoName();
        $globalref[0]->echoName();

        /* output:
        set on object creation
        set from outside */

        // let us see what is different with $bar2 and $globalref[1]
        $bar2->setName('set from outside');

        // luckily they are not only equyl, they are thesame variable
        // thus $bar2->Name and $globalref[1]->Name are the same too
        $bar2->echoName();
        $globalref[1]->echoName();

        /* output:
        set from outside
        set from outside */
```

Another final example, try to understand it.

```
class a {
    function a($i) {
        $this->value = $i;
        // try to figure out why we do not need a reference here
        $this->b = new b($this);
    }

    function createRef() {
        $this->c = new b($this);
    }

    function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->value;
        }
}


class b  {

    function b(&$a) {
        $this->a = &$a;
    }

   function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->a->value;
        }

}
```

```
// try to undestand why using a simple copy here would yield
// in an undesired result in the *-marked line
$a =& new a(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();

/*
output:
class a: 10
class b: 10
class b: 10
class a: 11
class b: 11
class b: 11
*/
```

Kapitola 14. References Explained

What References Are

References are a means in PHP to access the same variable content by different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names. The most close analogy is with Unix filenames and files - variable names are directory entries, while variable contents is the file itself. References can be thought of as hardlinking in Unix filesystem.

What References Do

PHP references allow you to make two variables to refer to the same content. Meaning, when you do:

```
$a =& $b
```

it means that `$a` and `$b` point to the same variable.

Poznámka: `$a` and `$b` are completely equal here, that's not `$a` is pointing to `$b` or vice versa, that's `$a` and `$b` pointing to the same place.

The same syntax can be used with functions, that return references, and with `new` operator (in PHP 4.0.4 and later):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

Poznámka: Unless you use the syntax above, the result of `$bar = new fooclass()` will not be the same variable as `$this` in the constructor, meaning that if you have used reference to `$this` in the constructor, you should use reference assignment, or you get two different objects.

The second thing references do is to pass variables by-reference. This is done by making a local variable in a function and a variable in the calling scope reference to the same content. Example:

```
function foo (&$var) {
    $var++;
}

$a=5;
foo ($a);
```

will make `$a` to be 6. This happens because in the function `foo` the variable `$var` refers to the same content as `$a`. See also more detailed explanations about passing by reference.

The third thing reference can do is return by reference.

What References Are Not

As said before, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var) {
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

What happens is that `$var` in foo will be bound with `$bar` in caller, but then it will be re-bound with `$GLOBALS["baz"]`. There's no way to bind `$bar` in the calling scope to something else using the reference

mechanism, since `$bar` is not available in the function foo (it is represented by `$var`, but `$var` has only variable contents and not name-to-value binding in the calling symbol table).

Passing by Reference

You can pass variable to function by reference, so that function could modify its arguments. The sytax is as follows:

```
function foo (&$var) {
    $var++;
}

$a=5;
foo ($a);
// $a is 6 here
```

Note that there's no reference sign on function call - only on function definition. Function definition alone is enough to correctly pass the argument by reference.

Following things can be passed by reference:

- Variable, i.e. `foo($a)`
- New statement, i.e. `foo(new foobar())`
- Reference, returned from a function, i.e.:

  ```
  function &bar()
  {
     $a = 5;
     return $a;
  }
  foo(bar());
  ```

  See also explanations about returning by reference.

Any other expression should not be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid:

```
function bar() // Note the missing &
{
        $a = 5;
        return $a;
}
foo(bar));

foo($a = 5) // Expression, not variable
foo(5) // Constant, not variable
```

These requirements are for PHP 4.0.4 and later.

Returning References

Returning by-reference is useful when you want to use a function to find which variable a reference should be bound to. When returning references, use this syntax:

```
function &find_var ($param) {
    ...code...
    return $found_var;
}

$foo =& find_var ($bar);
```

```
$foo->x = 2;
```

In this example, the property of the object returned by the `find_var` function would be set, not the copy, as it would be without using reference syntax.

> **Poznámka:** Unlike parameter passing, here you have to use `&` in both places - to indicate that you return by-reference, not a copy as usual, and to indicate that reference binding, rather than usual assignment, should be done for `$foo`.

Unsetting References

When you unset the reference, you just break the binding between variable name and variable content. This does not mean that variable content will be destroyed. For example:

```
$a = 1;
$b =& $a;
unset ($a);
```

won't unset `$b`, just `$a`.

Again, it might be useful to think about this as analogous to Unix **unlink** call.

Spotting References

Many syntax constructs in PHP are implemented via referencing mechanisms, so everything told above about reference binding also apply to these constructs. Some constructs, like passing and returning by-reference, are mentioned above. Other constructs that use references are:

`global` References

When you declare variable as **global $var** you are in fact creating reference to a global variable. That means, this is the same as:

```
$var =& $GLOBALS["var"];
```

That means, for example, that unsetting `$var` won't unset global variable.

`$this`

In an object method, `$this` is always reference to the caller object.

Část III. Vlastnosti

Kapitola 15. Error Handling

There are several types of errors and warnings in PHP. They are:

Tabulka 15-1. PHP error types

Value	Constant	Description	Note
1	E_ERROR	fatal run-time errors	
2	E_WARNING	run-time warnings (non fatal errors)	
4	E_PARSE	compile-time parse errors	
8	E_NOTICE	run-time notices (less serious than warnings)	
16	E_CORE_ERROR	fatal errors that occur during PHP's initial startup	PHP 4 only
32	E_CORE_WARNING	warnings (non fatal errors) that occur during PHP's initial startup	PHP 4 only
64	E_COMPILE_ERROR	fatal compile-time errors	PHP 4 only
128	E_COMPILE_WARNING	compile-time warnings (non fatal errors)	PHP 4 only
256	E_USER_ERROR	user-generated error message	PHP 4 only
512	E_USER_WARNING	user-generated warning message	PHP 4 only
1024	E_USER_NOTICE	user-generated notice message	PHP 4 only
	E_ALL	all of the above, as supported	

The above values (either numerical or symbolic) are used to build up a bitmask that specifies which errors to report. You can use the bitwise operators to combine these values or mask out certain types of errors. Note that only '|', '~', '!', and '&' will be understood within `php.ini`, however, and that no bitwise operators will be understood within `php3.ini`.

In PHP 4, the default error_reporting setting is `E_ALL & ~E_NOTICE`, meaning to display all errors and warnings which are not E_NOTICE-level. In PHP 3, the default setting is `(E_ERROR | E_WARNING | E_PARSE)`, meaning the same thing. Note, however, that since constants are not supported in PHP 3's `php3.ini`, the error_reporting setting there must be numeric; hence, it is `7`.

The initial setting can be changed in the ini file with the error_reporting directive, in your Apache `httpd.conf` file with the php_error_reporting (php3_error_reporting for PHP 3) directive, and lastly it may be set at runtime within a script by using the **error_reporting()** function.

Varování

When upgrading code or servers from PHP 3 to PHP 4 you should check these settings and calls to **error_reporting()** or you might disable reporting the new error types, especially E_COMPILE_ERROR. This may lead to empty documents without any feedback of what happened or where to look for the problem.

All PHP expressions can also be called with the "@" prefix, which turns off error reporting for that particular expression. If an error occurred during such an expression and the track_errors feature is enabled, you can find the error message in the global variable `$php_errormsg`.

Poznámka: The @ error-control operator prefix will not disable messages that are the result of parse errors.

> **Varování**
>
> Currently the @ error-control operator prefix will even disable error reporting for critical errors that will terminate script execution. Among other things, this means that if you use @ to suppress errors from a certain function and either it isn't available or has been mistyped, the script will die right there with no indication as to why.

Below we can see an example of using the error handling capabilities in PHP. We define a error handling function which logs the information into a file (using an XML format), and e-mails the developer in case a critical error in the logic happens.

Příklad 15-1. Using error handling in a script

```
<?php
// we will do our own error handling
error_reporting(0);

// user defined error handling function
function userErrorHandler ($errno, $errmsg, $filename, $linenum, $vars) {
    // timestamp for the error entry
    $dt = date("Y-m-d H:i:s (T)");

    // define an assoc array of error string
    // in reality the only entries we should
    // consider are 2,8,256,512 and 1024
    $errortype = array (
                1   =>  "Error",
                2   =>  "Warning",
                4   =>  "Parsing Error",
                8   =>  "Notice",
                16  =>  "Core Error",
                32  =>  "Core Warning",
                64  =>  "Compile Error",
                128 =>  "Compile Warning",
                256 =>  "User Error",
                512 =>  "User Warning",
                1024=>  "User Notice"
                );
    // set of errors for which a var trace will be saved
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errnumber>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";

    if (in_array($errno, $user_errors))
        $err .= "\t<vartrace>".wddx_serialize_value($vars,"Variables")."</vartrace>\n";
    $err .= "</errorentry>\n\n";

    // for testing
    // echo $err;

    // save to the error log, and e-mail me if there is a critical user error
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR)
        mail("phpdev@mydomain.com","Critical User Error",$err);
}


function distance ($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
```

```
        trigger_error("Incorrect parameters, arrays expected", E_USER_ERROR);
        return NULL;
    }

    if (count($vect1) != count($vect2)) {
        trigger_error("Vectors need to be of the same size", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}

$old_error_handler = set_error_handler("userErrorHandler");

// undefined constant, generates a warning
$t = I_AM_NOT_DEFINED;

// define some "vectors"
$a = array(2,3,"foo");
$b = array(5.5, 4.3, -1.6);
$c = array (1,-3);

// generate a user error
$t1 = distance($c,$b)."\n";

// generate another user error
$t2 = distance($b,"i am not an array")."\n";

// generate a warning
$t3 = distance($a,$b)."\n";

?>
```

This is just a simple example showing how to use the Error Handling and Logging functions.

See also **error_reporting()**, **error_log()**, **set_error_handler()**, **restore_error_handler()**, **trigger_error()**, **user_error()**

Kapitola 16. Tvorba a úpravy obrázků

PHP není omezeno na tvorbu pouze HTML výstupu. Může také vytvářet a upravovat obrázkové soubory různých formátů, včetně gif, png, jpg, wbmp a xpm. PHP může dokonce přímo posílat obrazové proudy do browseru. Na to budete potřebovat PHP zkompilované s GD knihovnou obrazových funkcí. GD a PHP mohou vyžadovat další knihovny v závislosti na obrazových formátech, se kterými chcete pracovat. GD přestala podporovat gif obrázky ve verzi 1.6.

Příklad 16-1. Tvorba PNG obrázků v PHP

```
<?php
    Header("Content-type: image/png");
    $string=implode($argv," ");
    $im = imageCreateFromPng("images/button1.png");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$string,$orange);
    ImagePng($im);
    ImageDestroy($im);
?>
```

Tento příklad by se volal ze stránky pomocí tagu podobného tomuto: <img src="button.php?text"> Skript button.php pak vezme řetězec "text", překryje jím základní obrázek, což je v tomto případě "images/button1.png" a zobrazí výsledný obrázek. Toto je vhodný způsob jak se vyhnout kreslení nového obrázku tlačítka pokaždé, když chcete změnit text tlačítka. Touto metodou se generují automaticky.

Kapitola 17. HTTP autentikace a PHP

Prostředky HTTP autentikace jsou v PHP přístupné pouze pokud PHP běží jako modul Apache, tudíž nejsou přístupné v CGI verzi. V PHP skriptu běžícím pod modulem Apache lze použít funkci **Header()** k odeslání zprávy "Authentication Required" klientskému browseru, což vyvolá zobrazení dialogového okna pro vložení uživatelského jména a hesla. Jakmile uživatel zadá jméno a heslo, URL obsahující tento PHP skript se zavolá znovu s proměnnými $PHP_AUTH_USER, $PHP_AUTH_PW and $PHP_AUTH_TYPE obsahujícími jméno, heslo a typ autentikace. V současnosti je podporována pouze "Basic" autentikace. Více informací viz funkce **Header()**.

Následující fragment kódu může posloužit jako ukázka vyžádání autentikace uživatele na stránce:

Příklad 17-1. Ukázka HTTP Autentikace

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Text, který se odešle, pokud uživatel zmáčkne tlačítko Cancel\n";
    exit;
  } else {
    echo "Ahoj $PHP_AUTH_USER.<P>";
    echo "Jako heslo jsi zadal $PHP_AUTH_PW.<P>";
  }
?>
```

Místo protého vytištění $PHP_AUTH_USER a $PHP_AUTH_PW byste asi chtěli ověřit platnost zadaného jména a hesla. Například dotazem v databázi nebo vyhledáním uživatele v dbm souboru.

Pozor na chybové browsery Internet Explorer. Zdá se, že jsou velice vybíravé, pokud jde o pořadí hlaviček. Zdá se, že odeslání hlavičky *WWW-Authenticate* před hlavičkou HTTP/1.0 401 zabírá.

Aby se zabránilo psaní skriptů odhalujících hesla na stránkách autentikovaných některým z tradičních externích mechanismů, PHP_AUTH proměnné se nevytvoří, pokud je pro tu kterou stránku zapnuta externí autentikace. V takovém případě můžete k identifikaci externě autentikovaného uživatele použít proměnnou $REMOTE_USER.

Všimněte si nicméně, že výše uvedené nezabrání krádežím hesel z autentikovaných URL osobou, která ovládá neautentikovanou URL na stejném serveru.

Jak Netscape, tak Internet Explorer po přijetí response kódu 401 vyprázdní autentikační cache současného realmu. Tak můžete uživatele v podstatě "odlogovat". Někteří lidé toho využívají k "vypršení" přihlášení nebo tvorbě odhlašovacího tlačítka.

Příklad 17-2. Ukázka HTTP autentikace vyžadující nové jméno a heslo

```
<?php
  function  authenticate()  {
      Header( "WWW-authenticate: basic  realm=\"Test  Authentication  System\"");
      Header( "HTTP/1.0  401  Unauthorized");
      echo  "K přístupu na tento zdroj musíte zadat platné ID a heslo\n";
      exit;
  }

  if(!isset($PHP_AUTH_USER)  ||  ($SeenBefore ==  1  &&  !str-
cmp($OldAuth,  $PHP_AUTH_USER))  )  {
    authenticate();
  }
  else  {
    echo  "Welcome:  $PHP_AUTH_USER<BR>";
    echo  "Old:  $OldAuth";
    echo  "<FORM  ACTION=\"$PHP_SELF\"  METHOD=POST>\n";
    echo  "<INPUT  TYPE=HIDDEN  NAME=\"SeenBefore\"  VALUE=\"1\">\n";
    echo  "<INPUT  TYPE=HIDDEN  NAME=\"OldAuth\"  VALUE=\"$PHP_AUTH_USER\">\n";
    echo  "<INPUT  TYPE=Submit  VALUE=\"Re  Authenticate\">\n";
    echo  "</FORM>\n";

}
```

```
?>
```

Podle standardu HTTP Basic authentication se toto chování nevyžaduje, takže byste na to nikdy neměli spoléhat. Pokusy s Lynxem ukázaly, že Lynx po přijetí response kódu 401 nevyprázdní autentikační údaje, takže po stisknutí back a forward se znovu ukáže požadovaný zdroj (pokud se nezměnily požadavky na údaje).

Dále si všimněte, že tato vlastnost při použití IIS serveru a CGI verze PHP díky omezením IIS nefunguje.

Kapitola 18. Cookies

PHP transparentně podporuje HTTP cookies. Cookies jsou mechanismem na ukládání dat ve vzdáleném browseru a tudíž sledování nebo identifikaci vracejících se uživatelů. Cookies můžete nastavovat pomocí funkce **setcookie()**. Cookies jsou součástí HTTP hlavičky, tudíž **setcookie()** se musí volat před odesláním výstupu do browseru. To je stejné omezení, jako má funkce **header()**.

Všechny cookies přijaté od klienta se automaticky stávají PHP proměnnou stejně jako u GET a POST dat. Pokud chcete přiřadit jednomu cookie více hodnot, přidejte *[]* na konec jména cookie. Více detailů viz funkce **setcookie()**.

Kapitola 19. Handling file uploads

POST method uploads

PHP is capable of receiving file uploads from any RFC-1867 compliant browser (which includes Netscape Navigator 3 or later, Microsoft Internet Explorer 3 with a patch from Microsoft, or later without a patch). This feature lets people upload both text and binary files. With PHP's authentication and file manipulation functions, you have full control over who is allowed to upload and what is to be done with the file once it has been uploaded.

Note that PHP also supports PUT-method file uploads as used by Netscape Composer and W3C's Amaya clients. See the PUT Method Support for more details.

A file upload screen can be built by creating a special form which looks something like this:

Příklad 19-1. File Upload Form

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

The _URL_ should point to a PHP file. The MAX_FILE_SIZE hidden field must precede the file input field and its value is the maximum filesize accepted. The value is in bytes.

In PHP 3, the following variables will be defined within the destination script upon a successful upload, assuming that register_globals is turned on in `php3.ini`. If track_vars is turned on, they will also be available in PHP 3 within the global array `$HTTP_POST_VARS`. Note that the following variable names assume the use of the file upload name 'userfile', as used in the example above:

- `$userfile` - The temporary filename in which the uploaded file was stored on the server machine.
- `$userfile_name` - The original name or path of the file on the sender's system.
- `$userfile_size` - The size of the uploaded file in bytes.
- `$userfile_type` - The mime type of the file if the browser provided this information. An example would be "image/gif".

Note that the "$userfile" part of the above variables is whatever the name of the INPUT field of TYPE=file is in the upload form. In the above upload form example, we chose to call it "userfile"

In PHP 4, the behaviour is slightly different, in that the new global array `$HTTP_POST_FILES` is provided to contain the uploaded file information. This is still only available if track_vars is turned on, but track_vars is always turned on in versions of PHP after PHP 4.0.2.

The contents of `$HTTP_POST_FILES` are as follows. Note that this assumes the use of the file upload name 'userfile', as used in the example above:

`$HTTP_POST_FILES['userfile']['name']`

The original name of the file on the client machine.

`$HTTP_POST_FILES['userfile']['type']`

The mime type of the file, if the browser provided this information. An example would be `"image/gif"`.

`$HTTP_POST_FILES['userfile']['size']`

The size, in bytes, of the uploaded file.

`$HTTP_POST_FILES['userfile']['tmp_name']`

The temporary filename of the file in which the uploaded file was stored on the server.

Files will by default be stored in the server's default temporary directory, unless another location has been given with the upload_tmp_dir directive in `php.ini`. The server's default directory can be changed by setting the environment variable TMPDIR in the environment in which PHP runs. Setting it using **putenv()** from within a PHP script will not work. This environment variable can also be used to make sure that other operations are working on uploaded files, as well.

Příklad 19-2. Validating file uploads

The following examples are for versions of PHP 3 greater than 3.0.16, and versions of PHP 4 greater than 4.0.2. See the function entries for **is_uploaded_file()** and **move_uploaded_file()**.

```
<?php
if (is_uploaded_file($userfile)) {
    copy($userfile, "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename ’$userfile’.";
}
/* ...or... */
move_uploaded_file($userfile, "/place/to/put/uploaded/file");
?>
```

For earlier versions of PHP, you’ll need to do something like the following.

Poznámka: This will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var(’upload_tmp_dir’)) {
        $tmp_file = dirname(tempnam(”, ”));
    }
    $tmp_file .= ’/’ . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace(’/+’, ’/’, $tmp_file) == $filename);
}

if (is_uploaded_file($userfile)) {
    copy($userfile, "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename ’$userfile’.";
}
?>
```

The PHP script which receives the uploaded file should implement whatever logic is necessary for determining what should be done with the uploaded file. You can for example use the `$file_size` variable to throw away any files that are either too small or too big. You could use the `$file_type` variable to throw away any files that didn’t match a certain type criteria. Whatever the logic, you should either delete the file from the temporary directory or move it elsewhere.

The file will be deleted from the temporary directory at the end of the request if it has not been moved away or renamed.

Common Pitfalls

The MAX_FILE_SIZE item cannot specify a file size greater than the file size that has been set in the upload_max_filesize in the PHP 3.ini file or the corresponding php3_upload_max_filesize Apache .conf directive. The default is 2 Megabytes.

Not validating which file you operate on may mean that users can access sensitive information in other directories.

Please note that the CERN httpd seems to strip off everything starting at the first whitespace in the content-type mime header it gets from the client. As long as this is the case, CERN httpd will not support the file upload feature.

Uploading multiple files

It is possible to upload multiple files simultaneously and have the information organized automatically in arrays for you. To do so, you need to use the same array submission syntax in the HTML form as you do with multiple selects and checkboxes:

Poznámka: Support for multiple file uploads was added in version 3.0.10.

Příklad 19-3. Uploading multiple files

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

When the above form is submitted, the arrays `$userfile`, `$userfile_name`, and `$userfile_size` will be formed in the global scope (as well as in $HTTP_POST_FILES ($HTTP_POST_VARS in PHP 3)). Each of these will be a numerically indexed array of the appropriate values for the submitted files.

For instance, assume that the filenames `/home/test/review.html` and `/home/test/xwp.out` are submitted. In this case, `$userfile_name[0]` would contain the value `review.html`, and `$userfile_name[1]` would contain the value `xwp.out`. Similarly, `$userfile_size[0]` would contain `review.html`'s filesize, and so forth.

`$userfile['name'][0]`, `$userfile['tmp_name'][0]`, `$userfile['size'][0]`, and `$userfile['type'][0]` are also set.

PUT method support

PHP provides support for the HTTP PUT method used by clients such as Netscape Composer and W3C Amaya. PUT requests are much simpler than a file upload and they look something like this:

```
PUT /path/filename.html HTTP/1.1
```

This would normally mean that the remote client would like to save the content that follows as: /path/filename.html in your web tree. It is obviously not a good idea for Apache or PHP to automatically let everybody overwrite any files in your web tree. So, to handle such a request you have to first tell your web server that you want a certain PHP script to handle the request. In Apache you do this with the *Script* directive. It can be placed almost anywhere in your Apache configuration file. A common place is inside a <Directory> block or perhaps inside a <Virtualhost> block. A line like this would do the trick:

```
Script PUT /put.php3
```

This tells Apache to send all PUT requests for URIs that match the context in which you put this line to the put.php3 script. This assumes, of course, that you have PHP enabled for the .php3 extension and PHP is active.

Inside your put.php3 file you would then do something like this:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

This would copy the file to the location requested by the remote client. You would probably want to perform some checks and/or authenticate the user before performing this file copy. The only trick here is that when PHP sees a

PUT-method request it stores the uploaded file in a temporary file just like those handled but the POST-method. When the request ends, this temporary file is deleted. So, your PUT handling PHP script has to copy that file somewhere. The filename of this temporary file is in the $PHP_PUT_FILENAME variable, and you can see the suggested destination filename in the $REQUEST_URI (may vary on non-Apache web servers). This destination filename is the one that the remote client specified. You do not have to listen to this client. You could, for example, copy all uploaded files to a special uploads directory.

Kapitola 20. Použití vzdálených souborů

Pokud při konfiguraci PHP aktivujete podporu "URL fopen wrapper" (standardně je zapnutá, ledaže pro configure explicitně zadáte `-disable-url-fopen-wrapper` příznak (verze do 4.0.3), nebo (u novějších verzí) nastavíte *`allow_url_fopen`* v php.ini na off), můžete ve voláních většiny funkcí, které očekávají jako argument název souboru (včetně **require()** a **include()**) uvést HTTP nebo FTP URL.

> **Poznámka:** Na Windows nelze používat vzdálené soubory v **include()** a **require()** výrazech.

Můžete například otevřít soubor na vzdáleném web serveru, vyseparovat z výstupu data, která potřebujete, a tato data potom použít v dotazu na databázi, nebo je prostě začlenit do výstupu stylem odpovídajícím zbytku vaší web site.

Příklad 20-1. Získání názvu vzdálené stránky

```
<?php
$file = fopen ("http://www.php.net/", "r");
if (!$file) {
    echo "<p>Nelze otevřít vzdálený soubor.\n";
    exit;
}
while (!feof ($file)) {
    $line = fgets ($file, 1024);
    /* Toto bude fungovat pouze pokud jsou tagy a název na jedné řádce */
    if (eregi ("<title>(.*)</title>", $line, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

Pokud se připojíte jako uživatel s dostatečnými právy, a daný soubor už neexistuje, můžete data také ukládat po FTP. Pokud se chcete připojit jako jiný uživatel než 'anonymous', musíte v URL udat uživatelské jméno (a pravděpodobně i heslo), např. 'ftp://uzivatel:heslo@ftp.example.com/path/to/file'. (Pro přístup k souborům přes HTTP, které vyžadují Basic authentication, můžete použít stejnou syntaxi.)

Příklad 20-2. Uložení dat na vzdáleném serveru

```
<?php
$file = fopen ("ftp://ftp.php.net/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Nelze otevřít vzdálený soubor pro zápis.\n";
    exit;
}
/* Zapíšeme data. */
fputs ($file, "$HTTP_USER_AGENT\n");
fclose ($file);
?>
```

> **Poznámka:** Z výše uvedeného příkladu by vás mohlo napadnout využít tuto techniku k zápisu do vzdáleného logu, ale jak už bylo zmíněno výše, pomocí URL fopen() wrapperu můžete zapisovat pouze do nového souboru. Pokud máte zájem o distribuované logování, podívejte se na **syslog()**.

Kapitola 21. Obsluha spojení

Poznámka: Následující text platí pro verzi 3.0.7 a vyšší.

Stav spojení se v PHP interně sleduje. Jsou tři možné stavy:

- 0 - NORMAL (normální)
- 1 - ABORTED (zrušeno)
- 2 - TIMEOUT (vypršel časový limit)

Při normálním běhu PHP skriptu je aktivní stav NORMAL. Pokud se klient odpojí, nastaví se příznak ABORTED. K odpojení vzdáleného klienta typicky dochází, když uživatel zmáčkne tlačítko STOP. Pokud se dosáhne časového limitu (viz **set_time_limit()**), nastaví se stavový příznak TIMEOUT.

Můžete se rozhodnout jestli chcete, aby odpojení klienta způsobilo předčasné ukončení vašeho skriptu. Někdy je užitečné nechat skripty doběhnout do konce, přestože není vzdáleného browseru, který by přijímal výstup. Výchozí chování je nicméně takové, že při odpojení vzdáleného klienta dojde k ukončení běhu skriptu. Toto chování se dá změnit skrze konfigurační direktivu ignore_user_abort v php3.ini, odpovídající direktivu php3_ignore_user_abort v .conf souboru Apache, či funkci **ignore_user_abort()**. Pokud nedáte PHP pokyn ignorovat odpojení uživatele a ten se odpojí, váš skript se ukončí. Výjimkou je, pokud máte pomocí **register_shutdown_function()** zaregistrovanou funkci pro provedení při ukončení skriptu. V tom případě, pokud vzdálený uživatel zmáčkne tlačítko STOP, při dalším pokusu tohoto skriptu odeslat výstup PHP detekuje, že spojení bylo zrušeno, a zavolá se funkce zaregistrovaná pro provedení při ukončení skriptu. Tato funkce se zavolá také na konci běhu skriptu končícím normálně, takže pokud chcete po zrušeném spojení udělat něco jiného, můžete použít **connection_aborted()**. Tato funkce vrátí `true`, pokud bylo spojení zrušeno.

Váš skript může také ukončit vestavěný čítač času. Výchozí časový limit je 30 sekund. To se dá změnit max_execution_time direktivou v phpš.ini nebo odpovídající php3_max_execution_time direktivou v .conf souboru Apahe, či voláním funkce **set_time_limit()**. Když čítač času doběhne, skript se ukončí, a jako ve výše uvedeném případě uživatelského odpojení, pokud je zaregistrovaná funkce pro provedení při ukončení skriptu, tato se zavolá. Uvnitř této funkce můžte zkontrolovat, jestli její zavolání způsobilo doběhnutí čítače času zavoláním funkce **connection_timeout()**. Tato funkce vrátí `true`, pokud volání funkce registrované pro provedení při ukončení skriptu způsobilo doběhnutí čítače času.

Skutečností hodnou povšimnutí je, že stavy ABORTED a TIMEOUT mohou být aktivní současně. Možné je to v případě, že nařídíte PHP ignorovat odpojení uživatee. PHP i tak bude vědět, že uživatel přerušil spojení, ale skript poběží dál. Pokud potom dosáhne časového limitu, bude ukončen, a zavolá se vaše funkce pro provedení při ukončení skriptu, pokud existuje. V tomto okamžiku zjistíte, že jak **connection_timeout()**, tak **connection_aborted()** vracejí `true`. Oba stavy můžete zkontrolovat jediným voláním funkce **connection_status()**. Tato funkce vrací bitové pole aktivních stavů. Takže například, pokud jsou aktivní oba tyto stavy, vrátí 3.

Kapitola 22. Persistentní databázová spojení

Trvalá spojení jsou SQL spojení, která se nezavírají na konci průběhu skriptu. Při požadavku na trvalé spojení PHP nejdříve zkontroluje, jestli už neexistuje identické spojení (které zůstalo otevřeno z dřívějška) - a pokud existuje, použije ho. Pokud neexistuje, PHP ho otevře. "Identické" spojení je spojení, které bylo otevřeno se stejným serverem, uživatelským jménem a heslem (pokud je zadáte).

Lidé, kteří nejsou důkladně obeznámeni se způsobem, jakým web servery fungují a distribuují zátěž mohou pokládat trvalá spojení za něco čím nejsou. Zvláště *neumožňují* otvírání "uživatelských sessions" na stejném SQL spojení, *neumožňují* efektivní tvorbu transakcí, a neumožňují spoustu dalších věcí. Dokonce, aby bylo opravdu a dúkladně jasno, vám trvalá spojení nedají *žádnou* funkcionalitu, která by nebyla možná s jejich netrvalými protějšky.

Proč?

To je dáno způsobem, jakým fungují webové servery. Jsou tři způsoby, jakými váš web server může využít PHP ke generování webových stránek.

První metodou je použít PHP jako CGI "obal". V tomto režimu se vytváří a ničí jedna instance PHP interpretru pro každý požadavek (na PHP strnánku) na vašem web serveru. Protože je znična po obsloužení požadavku, všechny zdroje, které získá (jako třeba spojení s databázovým serverem) jsou při jejím zničení zavřeny. V tomto případě pokusem o použití trvalých spojení nic nezískáte - prostě nevydrží.

Druhou, a nejpopulárnější, metodou, je provozovat PHP jako modul v multiprocesním web serveru, což je množina, která v současnosti obsahuje pouze Apache. Multiprocesní web serveru má typicky jeden proces (rodiče), který řídí skupinu procesů (svých dětí), které dělají vlastní práci - servírují stránky. Každý požadavek, který přijde od klienta, je obsloužen jedním z dětí, které právě neobsluhuje jiného klienta. To znamená, že když stejný klient vznese další požadavek na stejný server, tento může být obsloužen jiným dětským procesem než ten první. Trvalá spojení zajišťují, aby se každý dětský proces musel na váš SQL server přihlásit pouze při prvním odeslání stránky, která takové spojení využívá. Když spojení s SQL serverem vyžaduje další stránka, může použít spojení, které toto dítě otevřelo už dříve.

Poslední metodou je použít PHP jako plugin v multithreadovém web serveru. To je v současností pouhá teorie - PHP ještě nefunguje jako plagin v žádném multithreadovém web serveru. Pracuje se na podpoře ISAPI, WSAPI a NSAPI (na Windows), což umožní používat PHP jako plugin v multithreadových serverech jako Netscape FastTrack, Microsoft Internet Information Server (IIS), a O’Reilly’s WebSite Pro. Až k tomu dojde, chování bude v podstatě stejné jako u multiprocesním modelu popsaném dříve.

Pokud trvalá spojení neposkytují žádnou přidanou funkcionalitu, k čemu jsou dobrá?

Odpověď na tuto otázku je velmi jednoduchá - efektivita. Trvalá spojení jsou dobrá, pokud má tvorba spojení s vaším SQL serverem vysokou režii. Reálná výše této režie záleží na mnoha faktorech. Například jaký je to typ databáze, jestli sídlí na stejném počítači jako váš webserver, jak zatížený je stroj, na kterém váš SQL server běží a tak dále. Pointa je, že pokud je spojovací režie vysoká, trvalá spojení vám znatelně pomohou. Umožní dětskému procesu připojit se pouze jednou za celý jeho životní cyklus místo každého zpracování stránky, která vyžaduje spojení s SQL serverem. To znamená, že každé dítě, které otevřelo trvalé spojení, bude mít otevřené vlastní trvalé spojení se serverem. Pokud například máte 20 dětských procesů, které spustily skript, který otevřel trvalé spojení s vaším SQL serverem, máte 20 nezávislých spojení s SQL serverem, po jednom z každého dítěte.

Všimněte si nicméně, že to může mít nevýhody, pokud používate databázi s omezeným počtem připojení, který trvalá spojení dětí překročí. Pokud má vaše databáze limit 16 současných připojení, a v rušném okamžiku se pokusí připojit 17 dětských procesů, jednomu se to nepodaří. Pokud máte ve svých skriptech chyby, které brání zavírání spojení (např. nekonečné smyčky), databáze s pouhými 32 spojeními bude brzy zaplavena. Vyhledejte si v dokumentaci vaší databáze informace o obsluze opuštěných nebo nečinných spojení.

Důležitý souhrn. Trvalá spojení byla navržena tak, aby odpovídala jedna k jedné normálním spojením. To znamená, že byste *vždy* měli být schopni nahradit trvalá spojení netrvalými beze změny fungování vašeho skriptu. *Může* to (a pravděpodobně bude) mít vliv na efektivitu tohoto skriptu, ale ne jeho chování!

Část IV. Reference Funkcí

I. Funkce specifické pro Apache

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4)

Provádí částečný požadavek na zadanou URI a vrací všechno info o ní

```
class apache_lookup_uri (string filename)
```

Tato funkce provádí částečný požadavek na URI. Jde jen tak daleko, aby získala všechny důležité informace o daném zdroji a vrací tyto informace ve třídě. Vlastnosti této třídy jsou:

status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Poznámka: Apache_lookup_uri() pouze, pokud je PHP nainstalováno jako modul Apache.

apache_note (PHP 3>= 3.0.2, PHP 4)

Získává a nastavuje poznámky požadavku u Apache.

```
string apache_note (string note_name [, string note_value])
```

Apache_note() je funkce specifická pro Apache, která získává a nastavuje hodnoty v tabulce `poznámek` požadavku. Při volání s jedním argumentem vrací současnou hodnotu poznámky `note_name`. Při volání se dvěma argumenty nastavuje hodnotu poznámky `note_name` na `note_value`, a vrací předchozí hodnotu poznámky `note_name`.

getallheaders (PHP 3, PHP 4)

Získává všechny hlavičky HTTP požadavku

```
array getallheaders (void)
```

Tato funkce vrací asociativní pole všech HTTP hlaviček v současném požadavku.

Poznámka: Hodnotu běžných CGI proměnných můžete také získat tím, že je přečtete z prostředí, což funguje, ať používáte PHP jako modu Apache nebo ne. Pokud chcete vidět seznam všech systémových proměnných definovaných tímto způsobem, použijte **phpinfo()**.

Příklad 1. getallheaders() Example

```
$headers = getallheaders();
while (list ($header, $value) = each ($headers)) {
    echo "$header: $value<br>\n";
}
```

Tento příklad zobrazí všechny hlavičky současného požadavku.

Poznámka: Getallheaders() je v současnosti podporována jen když PHP běží jako modul Apache.

virtual (PHP 3, PHP 4)

Provádí sub-požadavek Apache

```
int virtual (string filename)
```

Virtual() je funkce specifická pro Apache ekvivalentní s <!–#include virtual...–> v mod_include. Provádí sub-požadavek Apache. Je užitečná pro vkládání CGI skriptů nebo .shtml souborů, nebo čehokoliv jiného, co má Apache parsovat. CGI skripty musí generovat platné CGI hlavičky. To znamená, že musí přinejmenším generovat Content-type hlavičku. Pro PHP soubory musíte použít **include()** nebo **require()**; **virtual()** se nedá použít k vložení dokumentu, který je sám PHP skriptem.

ascii2ebcdic (PHP 3>= 3.0.17)

Překládá řetězec z ASCII do EBCDIC

```
int ascii2ebcdic (string ascii_str)
```

ascii2ebcdic() je funkce specifická pro Apache dostupná pouze na operačních systémech založených na EBCDIC (OS/390, BS2000). Překládá (binárně bezpečně) řežězec v ASCII kódování *ascii_str* na jeho ekvivalentní EBCDIC reprezentaci, a vrací výsledek.

Viz také opačnou funkci **ebcdic2ascii()**

ebcdic2ascii (PHP 3>= 3.0.17)

Překládá řetězec z EBCDIC do ASCII

```
int ebcdic2ascii (string ebcdic_str)
```

ebcdic2ascii() je funkce specifická pro Apache dostupná pouze na operačních systémech založených na EBCDIC (OS/390, BS2000). Překládá (binárně bezpečně) řežězec v kódování EBCDIC *`ascii_str`* na jeho ekvivalentní ASCII reprezentaci, a vrací výsledek.

Viz také opačnou funkci **ascii2ebcdic()**

II. Funkce pro práci s poli

Tyto funkce vám umožňují manipulovat a interagovat různými způsoby s poli. Pole jsou nezbytná pro ukládání a práci se sadami proměnných.

Podporována jsou jednoduchá a vícerozměrná pole; vytvářet se dají uživatelsky i jako výstup funkce. Existují databázové funkce na plnění polí výsledky databázových dotazů, a několik dalších funkcí vrací pole.

Viz také: **is_array()**, **explode()**, **implode()**, **split()** a **join()**.

array (unknown)

Vytvořit pole

```
array array ([mixed ...])
```

Vrací pole argumentů. Argumentům může být přiřazen index pomocí operátoru =>.

> **Poznámka: array()** je jazykový konstrukt používaný k reprezentaci polí, nikoliv běžná funkce.

Syntaxe "index => hodnota", s čárko jako oddělovačem, definuje indexy a hodnoty. Index může být řetězec nebo číslo. Pokud se index vynechá, automaticky se generuje číselný index začínající na 0. Pokud je index integer, další generovaný index bude nejvyšší celočíselný index + 1. Pozn.: pokud jsou definovány dva identické indexy, první se přepíše posledním.

Následující ukázka demonstruje jak vytvořit dvourozměrné pole, jak určit klíče v asociativních polích, a jak přeskakovat číselné indexy v normálních polích.

Příklad 1. Ukázka array()

```
$fruits = array (
    "fruits"  => array ("a"=>"orange", "b"=>"banana", "c"=>"apple"),
    "numbers" => array (1, 2, 3, 4, 5, 6),
    "holes"   => array ("first", 5 => "second", "third")
);
```

Příklad 2. Automatický index a array()

```
$array = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
print_r($array);
```

výstup bude následující:

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Index 3 je definován dvakrát, a podrží si poslední hodnotu 13. Index 4 je definován po indexu 8 a další generovaný index (hodnota 19) je 9, protože nejvyšší index byl 8.

Tato ukázka vytvoří pole číslované od 1.

Příklad 3. Index začínající 1 s array()

```
        $firstquarter  = array(1 => 'January', 'February', 'March');
        print_r($firstquarter);
```

toto bude výstup:

```
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
```

Viz také: **list()**.

array_count_values (PHP 4 >= 4.0b4)

Spočítat všechny hodnoty v poli

```
array array_count_values (array input)
```

array_count_values() vrací pole používající hodnoty z *input* jako klíče a jejich četnosti v *input* jako hodnoty.

Příklad 1. Ukázka array_count_values()

```
$array = array (1, "hello", 1, "world", "hello");
array_count_values ($array); // vrací array(1=>2, "hello"=>2, "world"=>1)
```

array_diff (PHP 4 >= 4.0.1)

Spočítat rozdíl polí

```
array array_diff (array array1, array array2 [, array ...])
```

array_diff() vrací pole obsahující všechny hodnoty z *array1*, které se nevyskytují v žádném z dalších argumentů. Klíče jsou zachovány.

Příklad 1. Ukázka array_diff()

```
$array1 = array ("a" => "green", "red", "blue");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_diff ($array1, $array2);
```

`$result` obsahuje `array ("blue");`

Viz také: **array_intersect()**.

array_flip (PHP 4 >= 4.0b4)

Prohodit klíče a hodnoty pole

```
array array_flip (array trans)
```

array_flip() pole s prohozenými klíči a hodnotami.

Příklad 1. Ukázka array_flip()

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

array_intersect (PHP 4 >= 4.0.1)

Spočítat průnik polí

```
array array_intersect (array array1, array array2 [, array ...])
```

array_intersect() vrací pole obsahující všechny hodnoty z *array1*, které se vyskytují ve všech argumentech. Klíče jsou zachovány.

Příklad 1. Ukázka array_intersect()

```
$array1 = array ("a" => "green", "red", "blue");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_intersect ($array1, $array2);
```

`$result` obsahuje `array ("a" => "green", "red");`

Viz také: **array_diff()**.

array_keys (PHP 4)

Vrátit všechny klíče pole

```
array array_keys (array input [, mixed search_value])
```

array_keys() vrací klíče, numerické i textové, z pole *input*.

Pokud je přítomen volitelný argument *search_value*, vrací pouze klíče této hodnoty. Jinak vrací všechny klíče z pole *input*.

Příklad 1. Ukázka array_keys()

```
$array = array (0 => 100, "color" => "red");
array_keys ($array);       // vrací array (0, "color")

$array = array ("blue", "red", "green", "blue", "blue");
array_keys ($array, "blue");  //  vrací array (0, 3, 4)

$array = array ("color" => array("blue", "red", "green"), "size" => ar-
ray("small", "medium", "large"));
array_keys ($array);  //  vrací array ("color", "size")
```

Poznámka: Tato funkce byla přidána v PHP 4, dále je uvedena implementace pro ty, kteří stále používají PHP 3.

Příklad 2. Implementace array_keys() pro uživatele PHP 3

```
function array_keys ($arr, $term="") {
```

```
        $t = array();
        while (list($k,$v) = each($arr)) {
            if ($term && $v != $term)
                continue;
                $t[] = $k;
            }
            return $t;
    }
```

Viz také: **array_values()**.

array_merge (PHP 4)

Sloučit dvě nebo více polí

```
array array_merge (array array1, array array2 [, array ...])
```

array_merge() sloučí prvky dvou nebo více polí dohromady tak, že hodnoty každého pole se připojí na konec předchozího. Vrací výsledné pole.

Pokud mají vstupní pole stejný textový klíč, pozdější hodnota přepíše dřívější hodnotu. Pokud ale mají stejný číselný klíč, pozdější hodnota tu púvodní nepřepíše, ale připojí se k ní.

Příklad 1. Ukázka array_merge()

```
$array1 = array ("color" => "red", 2, 4);
$array2 = array ("a", "b", "color" => "green", "shape" => "trapezoid", 4);
array_merge ($array1, $array2);
```

Výsledné pole bude `array("color" => "green", 2, 4, "a", "b", "shape" => "trapezoid", 4)`.

Viz také: **array_merge_recursive()**.

array_merge_recursive (PHP 4 >= 4.0.1)

Rekurzivně sloučit dvě nebo více polí

```
array array_merge_recursive (array array1, array array2 [, array ...])
```

array_merge_recursive() sloučí prvky dvou nebo více polí tak, že hodnoty pole se připojí na konec předchozího pole. Vrací výsledné pole.

Pokud obsahují vstupní pole stejný textový klíč, hodnoty těchto klíčů se rekurzivně sloučí do pole tak, že pokud je jedna z hodnot sama pole, tato funkce ji také sloučí s odpovídající položkou z dalšího pole. Pokud ale tato pole mají stejný číselný klíč, pozdější hodnota nepřepíše tu dřívější, ale připojí se.

Příklad 1. Ukázka array_merge_recursive()

```
$ar1 = array ("color" => array ("favorite" => "red"), 5);
$ar2 = array (10, "color" => array ("favorite" => "green", "blue"));
$result = array_merge_recursive ($ar1, $ar2);
```

Výsledné pole bude `array ("color" => array ("favorite" => array ("red", "green"), "blue"), 5, 10)`.

Viz také: **array_merge()**.

array_multisort (PHP 4 >= 4.0b4)

Třídit více polí, nebo vícerozměrné pole

```
bool array_multisort (array ar1 [, mixed arg [, mixed ... [, array ...]]])
```

array_multisort() se dá využít k třídění několika polí najednou nebo k třídění vícerozměrného pole XXX according by one of more dimensions. Při třídění udržuje asociace klíčů.

Vstupní pole jsou manipulována jako sloupce tabulky, která se má třídit podle řádků - připomíná to funkcionalitu SQL klauzule ORDER BY. První pole je to, podle kterého se bude třídit. Řádky (hodnoty) v tomto poli that compare the same are sorted by the next input array, and so on.

Struktura argumentů této funkce je trochu neobvyklá, ale pružná. První argument musí být pole. Každý další argument může být buď pole nebo jeden z příznak z následujících seznamů:

Příznaky směru třídění:

- SORT_ASC - třídit vzestupně
- SORT_DESC - třídit sestupně

Příznaky typu třídění:

- SORT_REGULAR - porovnávat položky normálně
- SORT_NUMERIC - porovnávat položky číselně
- SORT_STRING - porovnávat položky jako řetězce

Po každém poli můžete specifikovat jeden příznak každého typu. Příznaky třídění specifikované po každém poli platí pouze pro toto pole - pro další pole se resetují na defaultní SORT_ASC a SORT_REGULAR.

Při úspěchu vrací `true`, při selhání `false`.

Příklad 1. Třídění více polí

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

V této ukázce bude po setřídění první pole obsahovat 10, "a", 100, 100. Druhé pole bude obsahovat 1, 1, 2, "3". Položky druhého pole odpovídající identickým položkám v prvním poli (100 a 100) byly také setříděny.

Příklad 2. Třídění vícerozměrného pole

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

V této ukázce bude po setřídění první pole obsahovat 10, 100, 100, "a" (bylo tříděno vzestupně jako řetězce) a druhé pole bude obsahovat 1, 3, "2", 1 (tříděno jako čísla, sestupně).

array_pad (PHP 4 >= 4.0b4)

Doplnit pole hodnotou na určenou délku

```
array array_pad (array input, int pad_size, mixed pad_value)
```

array_pad() vrací kopii pole *input* doplněnou na velikost *pad_size* hodnotou *pad_value*. Pokud je *pad_size* kladná, pole je doplněno zprava, pokud je negativní, zleva. Pokud je absolutní hodnota *pad_size* menší nebo rovna velikosti *input*, k doplnění nedojde.

Příklad 1. Ukázka array_pad()

```
$input = array (12, 10, 9);

$result = array_pad ($input, 5, 0);
// výsledek je array (12, 10, 9, 0, 0)

$result = array_pad ($input, -7, -1);
// výsledek je array (-1, -1, -1, -1, 12, 10, 9)

$result = array_pad ($input, 2, "noop");
// nedoplněno
```

array_pop (PHP 4)

Odstranit prvek z konce pole

```
mixed array_pop (array array)
```

array_pop() odstraní a vrátí poslední hodnotu pole *array*, čímž ho zkrátí o jeden prvek.

Příklad 1. Ukázka Array_pop()

```
$stack = array ("orange", "apple", "raspberry");
$fruit = array_pop ($stack);
```

`$stack` má teď pouze dva prvky: "orange" a "apple", a `$fruit` obsahuje "raspberry".

Viz také: **array_push()**, **array_shift()** a **array_unshift()**.

array_push (PHP 4)

Přidat jeden nebo více prvků na konec pole

```
int array_push (array array, mixed var [, mixed ...])
```

array_push() připojuje předané proměnné na konec *array*. Délka *array* se zvětšuje o počet přidaných proměnných. Účinek je stejný jako:

```
$array[] = $var;
```

opakované pro každou *var*.

Vrací nový počet prvků v poli.

Příklad 1. Ukázka array_push()

```
$stack = array (1, 2);
array_push ($stack, "+", 3);
```

Výsledkem této ukázky by byl `$stack` obsahující 4 prvky: 1, 2, "+" a 3.

Viz také: **array_pop()**, **array_shift()** a **array_unshift()**.

array_rand (PHP 4 >= 4.0.0)

Vybrat náhodně jeden nebo více prvků pole

```
mixed array_rand (array input [, int num_req])
```

array_rand() je poměrně užitečná, když chcete z pole vybrat náhodně jednu nebo více hodnot. Přijímá pole *`input`* a volitelný argument *`num_req`*, který určuje, kolik položek chcete. Jeho defaultní hodnota je 1.

Pokud vybíráte pouze jednu položku, **array_rand()** vrací klíč náhodné položky. Jinak vrací pole klíčů náhodně vybraných položek. Takto můžete vybírat náhodně hodnoty i klíče.

Nezapomeňte inicializovat generátor náhodných čísel pomocí **srand()**.

Příklad 1. Ukázka array_rand()

```
srand ((double) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand ($input, 2);
print $input[$rand_keys[0]]."\n";
print $input[$rand_keys[1]]."\n";
```

array_reverse (PHP 4 >= 4.0b4)

Vrátit pole s prvky v opačném pořadí

```
array array_reverse (array array [, bool preserve_keys])
```

array_reverse() takes input *`array`* and returns a new array with the order of the elements reversed, preserving the keys if *`preserve_keys`* is true.

Příklad 1. Array_reverse() example

```
$input = array ("php", 4.0, array ("green", "red"));
$result = array_reverse ($input);
$result_keyed = array_reverse ($input, true);
```

`$result` teď obsahuje `array (array ("green", "red"), 4.0, "php")`. Ale `$result2[0]` je stále `"php"`.

Poznámka: Druhý argument byl přidán v PHP 4.0.3.

array_shift (PHP 4)

Odstranit prvek ze začátku pole

```
mixed array_shift (array array)
```

array_shift() vrací první položku *array* a odstraní ji, čímž zkrátí *array* o jeden prvek a ostatní posune dolů.

Příklad 1. Ukázka array_shift()

```
$args = array ("-v", "-f");
$opt = array_shift ($args);
```

$args teď má jeden prvek: "-f" a $opt je "-v".

Viz také: **array_unshift()**, **array_push()** a **array_pop()**.

array_slice (PHP 4)

Vytáhnout část pole

```
array array_slice (array array, int offset [, int length])
```

array_slice() vrací sekvenci prvků *array* určených argumenty *offset* a *length*.

Pokud je *offset* kladný, tato sekvence začne *offset* položek od začátku *array*. Pokud je *offset* záporný, tato sekvence začne tolik položek od konce *array*.

Pokud je *length* kladná, tato sekvence bude obsahovat tolik prvků. Pokud je *length* záporná, tato sekvence skončí tolik prvků od konce *array*. Pokud *length* vynecháte, tato sekvence bude obsahovat všechny prvky *array* od *offset* do konce.

Příklad 1. Ukázka array_slice()

```
$input = array ("a", "b", "c", "d", "e");

$output = array_slice ($input, 2);      // returns "c", "d", and "e"
$output = array_slice ($input, 2, -1);  // returns "c", "d"
$output = array_slice ($input, -2, 1);  // returns "d"
$output = array_slice ($input, 0, 3);   // returns "a", "b", and "c"
```

Viz také: **array_splice()**.

array_splice (PHP 4)

Odstranit část pole a nahradit ji něčím jiným

```
array array_splice (array input, int offset [, int length [, array replacement]])
```

array_splice() odstraňuje prvky pole *input* určené argumenty *offset* a *length*, a případně je nahrazuje prvky volitelného argumentu (pole) *replacement*.

Pokud je *offset* kladný, tato odstraněná část začne *offset* položek od začátku *array*. Pokud je *offset* záporný, začne tolik položek od konce *array*.

Pokud vynecháte *length*, **array_splice()** odstraní všechno od *offset* do konce pole. Pokud je *length* kladná, odstraní se právě tolik prvků. Pokud je *length* záporná, konec odstraněné části bude právě tolik prvků od konce pole. Tip: k odstranění všech prvků od *offset* do konce pole při současně určeném argumentu *replacement* použijte jako *length* `count($input)`.

Pokud zadáte *replacement* pole, odstraněné prvky se nahradí prvky tohoto pole. Pokud argumenty *offset* a *length* definovány tak, že se nic neodstraní, prvky pole *replacement* se vloží na místo určené argumentem *offset*. Tip: pokud je *replacement* jen jedna hodnota, není nutno ji umisťovat do `array()`, ledaže chcete, aby tato položka byla opravdu pole.

Následující volání jsou ekvivalentní:

```
array_push ($input, $x, $y)     array_splice ($input, count ($input), 0,
                                             array ($x, $y))
array_pop ($input)              array_splice ($input, -1)
array_shift ($input)            array_splice ($input, 0, 1)
array_unshift ($input, $x, $y)  array_splice ($input, 0, 0, array ($x, $y))
$a[$x] = $y                     array_splice ($input, $x, 1, $y)
```

Vrací pole odstraněných prvků.

Příklad 1. Ukázky array_splice()

```
$input = array ("red", "green", "blue", "yellow");

array_splice ($input, 2);      // $input is now array ("red", "green")
array_splice ($input, 1, -1);  // $input is now array ("red", "yellow")
array_splice ($input, 1, count($input), "orange");
                               // $input is now array ("red", "orange")
array_splice ($input, -1, 1, array("black", "maroon"));
                               // $input is now array ("red", "green",
                               //          "blue", "black", "maroon")
```

Viz také: **array_slice()**.

array_unique (PHP 4 >= 4.0.1)

Odstranit z pole duplicitní hodnoty

```
array array_unique (array array)
```

array_unique() přijímá pole *array* a vrací nové pole bez duplicitních hodnot. Klíče jsou zachovány.

Příklad 1. Ukázka array_unique()

```
$input = array ("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique ($input);
```

`$result` teď obsahuje `array ("a" => "green", "red", "blue");`.

array_unshift (PHP 4)

Připojit jeden nebo více prvků na začátek pole

```
int array_unshift (array array, mixed var [, mixed ...])
```

array_unshift() připojí předané prvky na začátek *array*. Všechny prvky jsou přidány jako celek, čili přidané prvky si zachovávají pořadí.

vrací nový počet prvků v *array*.

Příklad 1. Ukázka array_unshift()

```
$queue = array ("p1", "p3");
array_unshift ($queue, "p4", "p5", "p6");
```

$queue bude mít 5 prvků: "p4", "p5", "p6", "p1" a "p3".

Viz také: **array_shift()**, **array_push()** a **array_pop()**.

array_values (PHP 4)

Vrátit všechny hodnoty v poli

```
array array_values (array input)
```

array_values() vrací všechny hodnoty pole *input*.

Příklad 1. Ukázka array_values()

```
$array = array ("size" => "XL", "color" => "gold");
array_values ($array);    // vrátí array ("XL", "gold")
```

Poznámka: Tato funkce byla přidána v PHP 4, následuje implementace pro ty, kdo stále používají PHP 3.

Příklad 2. Implementace array_values() pro uživatele PHP 3

```
function array_values ($arr) {
    $t = array();
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
        return $t;
    }
}
```

array_walk (PHP 3>= 3.0.3, PHP 4)

Použít uživatelskou funkci na všechny prvky pole

```
int array_walk (array arr, string func, mixed userdata)
```

Aplikuje funkci *func* na všechny prvky pole *arr*. Funkci *func* se jako první argument předá hodnota a jako druhý klíč. Pokud je přítomen argument *userdata*, bude uživatelské funkci předán jako třetí argument.

Pokud *func* vyžaduje více než dva nebo tři argumenty (v závislosti na *userdata*), pro každé volání *func* z **array_walk()** se vygeneruje varování. Tato varování se dají potlačit přidáním znaku '@' před volání **array_walk()** nebo pomocí **error_reporting()**.

> **Poznámka:** Pokud *func* potřebuje pracovat přímo s daným polem, první argument *func* se musí předávat odkazem. Všechny změny těchto hodnot se pak promítnou přímo v *arr*.

> **Poznámka:** Druhý a třetí argument *func* byly přidány v PHP 4.0.
>
> V PHP 4 je třeba volat podle potřeby **reset()**, protože **array_walk()** sama vstupní pole neresetuje.

Příklad 1. Ukázka array_walk()

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");

function test_alter (&$item1, $key, $prefix) {
    $item1 = "$prefix: $item1";
}

function test_print ($item2, $key) {
    echo "$key. $item2<br>\n";
}

array_walk ($fruits, 'test_print');
reset ($fruits);
array_walk ($fruits, 'test_alter', 'fruit');
reset ($fruits);
array_walk ($fruits, 'test_print');
```

Viz také: **each()** a **list()**.

arsort (PHP 3, PHP 4)

Třídit pole sestupně se zachováním klíčů

```
void arsort (array array [, int sort_flags])
```

arsort() třídí pole tak, že indexy pole si zachovávají korelace s prvky, se kterými jsou asociovány. Toto je užitečné hlavně při třídění asociativních polí, kde je pořadí prvků signifikantní.

Příklad 1. Ukázka Arsort()

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
arsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Tato ukázka zobrazí:

```
fruits[a] = orange
fruits[d] = lemon
fruits[b] = banana
fruits[c] = apple
```

Ovoce bylo sestupně setříděno podle abecedy, a indexy asociované s jednotlivými prvky byly zachovány.

Chování třídění můžete upravit pomocí volitelného argumentu *sort_flags*, detaily viz **sort()**.

Viz také: **asort()**, **rsort()**, **ksort()** a **sort()**.

asort (PHP 3, PHP 4)

Třídit pole se zachováním indexů

```
void asort (array array [, int sort_flags])
```

asort() třídí pole tak, že si indexy zachovají corelace s prvky, se kterými jsou spojeny. To je užitečné hlavně při třídění asociativních polí, u kterých je pořadí prvků signifikantní.

Příklad 1. Ukázka Asort()

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
asort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Tato ukázka zobrazí:

```
fruits[c] = apple
fruits[b] = banana
fruits[d] = lemon
fruits[a] = orange
```

Ovoce bylo setříděno podle abecedy a indexy spojené s jednotlivými prvky byly zachovány.

Chování třídění můžete upravit pomocí volitelného argumentu *sort_flags*, detaily viz **sort()**.

Viz také: **arsort()**, **rsort()**, **ksort()** a **sort()**.

compact (PHP 4)

Vytvořit pole obsahující proměnné a jejich hodnoty

```
array compact (mixed varname [, mixed ...])
```

compact() přijímá proměnný počet argumentů. Každý argument může být buď řetězec obsahující název proměnné nebo pole názvů proměnných. Toto pole může také obsahovat pole názvů proměnnýchů; **compact()** je rekurzivně zpracuje.

Pro každý z řetězců **compact()** vyhledá v aktivní symbolové tabulce proměnnou tohoto jména a přidá ji do výsledného pole tak, že název této proměnné se stane klíčem a obsah této proměnné hodnotou tohoto klíče. Stručně řečeno, dělá pravý opak toho, co **extract()**. Vrací pole obsahující všechny tyto proměnné.

Řetězce, které neobsahují názvy platných proměnných se přeskočí.

Příklad 1. Ukázka compact()

```
$city = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";

$location_vars = array ("city", "state");

$result = compact ("event", "nothing_here", $location_vars);
```

`$result` bude `array ("event" => "SIGGRAPH", "city" => "San Francisco", "state" => "CA")`.

Viz také: **extract()**.

count (PHP 3, PHP 4)

Spočítat prvky v proměnné

```
int count (mixed var)
```

Vrací počet prvků v *`var`*, která je typicky pole (jelikož všechno ostatní má jeden prvek).

Vrací `1`, pokud *`var`* není pole.

Vrací `0`, pokud *`var`* není inicializována.

> **Varování**
>
> **count()** vrací `0` pro proměnné, které nejsou inicializovány, ale vrací také `0` pro proměnné, které obsahují prázdné pole. Pokud potřebujete zjistit, jestli dotyčná proměnná existuje, použijte **isset()**.

Příklad 1. Ukázka count()

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count ($a);
//$result == 3
```

Viz také: **sizeof()**, **isset()** a **is_array()**.

current (PHP 3, PHP 4)

Vrátit současný prvek pole

```
mixed current (array array)
```

Každé pole má vnitřní ukazatel na jeho "současny" prvek, který se inicializuje na první prvek vložený do tohoto pole.

current() vrací prvek, na který tento interní ukazatel právě ukazuje. Nijak tento ukazatel nemění. Pokud teto vnitřní ukazatel ukazuje za konec seznamu prvků, **current()** returns `false`.

> **Varování**
>
> Pokud toto pole obsahuje prázdné prvky (0 nebo "", prázdný řetězec), tato funkce pro tyto prvky také vrátí `false`. Je proto nemožné určit pomocí **current()**, jestli jste opravdu na konci pole. To properly traverse an array that may contain empty elements, use the **each()** function.

Viz také: **end()**, **next()**, **prev()** a **reset()**.

each (PHP 3, PHP 4)

Vrací další klíč/hodnota pár z pole

```
array each (array array)
```

Vrací současný klíč/hodnota pár z pole *array* a posune interní ukazatel pole. Tento pár se vrací jako pole čtyř prvků s klíči *0*, *1*, *key* a *value*. Prvky *0* a *key* obsahují název klíče tohoto prvku pole a *1* a *value* obsahují hodnotu.

Pokud interní ukazatel pole ukazuje za konec tohoto pole, **each()** vrací `false`.

Příklad 1. Ukázky each()

```
$foo = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each ($foo);
```

`$bar` teď obsahuje následující klíč/hodnota páry:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each ($foo);
```

`$bar` teď obsahuje následující klíč/hodnota páry:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

each() se většinou používa s **list()** k průchodu polem, např. `$HTTP_POST_VARS`:

Příklad 2. Průchod `$HTTP_POST_VARS` pomocí each()

```
echo "Hodnoty odeslané metodou POST:<br>";
reset ($HTTP_POST_VARS);
while (list ($key, $val) = each ($HTTP_POST_VARS)) {
    echo "$key => $val<br>";
}
```

Poté, co proběhne **each()**, se interní ukazatel pole posune na další prvek pole nebo zůstane na posledním prvku pole, pokud dojde na konec.

Viz také: **key()**, **list()**, **current()**, **reset()**, **next()** a **prev()**.

end (PHP 3, PHP 4)

Nastavit vnitřní ukazatel pole na jeho poslední prvek

```
mixed end (array array)
```

end() posune vnitřní ukazatel pole *array* na jeho poslední prvek a vrací tento prvek.

Viz také: **current()**, **each()**, **end()**, **next()** a **reset()**.

extract (PHP 3>= 3.0.7, PHP 4)

Importovat proměnné z pole do symbolové tabulky

```
int extract (array var_array [, int extract_type [, string prefix]])
```

Tato funkce se používá k importu proměnných z pole do aktivní symbolové tabulky. Přijímá pole *var_array*; z klíčů vytváří názvy proměnných a z hodnot hodnoty těchto proměnných. Vytváří jednu proměnnou z každého klíč/hodnota páru (s ohledem na argumenty *extract_type* a *prefix*).

> **Poznámka:** Od PHP 4.0.5 tato funkce vrací počet extrahovaných proměnných.

extract() ověřuje, jestli všechny klíče tvoří platné názvy proměnných, a také jestli nekolidují s proměnnými existujícími v aktivní symbolové tabulce. Způsob, jakým se nakládá s neplatnými/numerickými klíči a kolizemi závisí na *extract_type*. Ten může mít jednu z následujících hodnot.

EXTR_OVERWRITE

Pokud existuje kolize, přepsat existující proměnnou.

EXTR_SKIP

Pokud existuje kolize, nepřepsat existující proměnnou.

EXTR_PREFIX_SAME

Pokud existuje kolize, předřadit před název nové proměnné *prefix*.

EXTR_PREFIX_ALL

Opatřit prefixem *prefix* všechny názvy proměnných. Od PHP 4.0.5 toto zahrnuje i číselné indexy.

EXTR_PREFIX_INVALID

Prefixem *prefix* opatřit pouze neplatné/číselné názvy proměnných. Tento příznak byl přidán v PHP 4.0.5.

Defaultní *extract_type* je EXTR_OVERWRITE.

Pozn.: *prefix* se vyžaduje pouze pokud je *extract_type* EXTR_PREFIX_SAME, EXTR_PREFIX_ALL nebo EXTR_PREFIX_INVALID. Pokud výsledný název (vč. prefixu) není platný název proměnné, nenaimportuje se do symbolové tabulky.

extract() vrací počet proměnných úspěšně naimportovaných do symbolové tabulky.

Možné využití **extract()** je import proměnných do symbolové tabulky z asociativního pole vráceného **wddx_deserialize()**.

Příklad 1. Ukázka Extract()

```
<?php

/* Předpokládejme, že $var_array je pole vrácené
   z wddx_deserialize */

$size = "large";
$var_array = array ("color" => "blue",
                    "size"  => "medium",
                    "shape" => "sphere");
extract ($var_array, EXTR_PREFIX_SAME, "wddx");

print "$color, $size, $shape, $wddx_size\n";

?>
```

Výše uvedená ukázka vytiskne:

```
blue, large, sphere, medium
```

`$size` se nepřepsala, protože bylo specifikováno EXTR_PREFIX_SAME, tudíž se vytvořila proměnná `$wddx_size`. Pokud by bylo zadáno EXTR_SKIP, nevytvořila by se ani `$wddx_size`. EXTR_OVERWRITE by způsobilo přepsání hodnoty `$size` na "medium", a EXTR_PREFIX_ALL by vytvořilo nové proměnné pojmenované `$wddx_color`, `$wddx_size` a `$wddx_shape`.

U PHP verzí nižších než 4.0.5 musíte použít asociativní pole.

Viz také: **compact()**.

in_array (PHP 4)

Vrátit `true`, pokud v poli existuje daná hodnota

```
bool in_array (mixed needle, array haystack, bool strict)
```

Hledá v *`haystack`* *`needle`* a pokud ji najde, vrací `true`, jinak `false`.

Pokud je třetí argument *`strict`* `true`, **in_array()** také kontroluje typ *`needle`* v *`haystack`*.

Příklad 1. Ukázka in_array()

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os)){
    print "Got Irix";
    }
```

Příklad 2. Ukázka in_array() s argumentem *`strict`*

```
// Výstup bude:
```

```
1.13 found with strict check
```

key (PHP 3, PHP 4)

Fetch a key from an associative array

```
mixed key (array array)
```

key() vrací index současného prvku pole.

Viz také: **current()** a **next()**.

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Třídit pole sestupně podle klíčů

```
int krsort (array array [, int sort_flags])
```

Třídí pole sestupně podle klíčů se zachování asociací indexů. To je užitečné hlavně při práci s asociativními poli.

Příklad 1. Krsort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key -> $val\n";
}
```

Tato ukázka vytiskne:

```
fruits[d] = lemon
fruits[c] = apple
fruits[b] = banana
fruits[a] = orange
```

Vlastnosti třídění lze upravit pomocí volitelného argumentu *sort_flags*, detaily viz **sort()**.

Viz také: **asort()**, **arsort()**, **ksort()**, **sort()**, **natsort()** a **rsort()**.

ksort (PHP 3, PHP 4)

Třídit pole podle klíčů

```
int ksort (array array [, int sort_flags])
```

Třídí pole podle klíčů se zachování asociací indexů. To je užitečné hlavně při práci s asociativními poli.

Příklad 1. Ksort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key -> $val\n";
}
```

Tato ukázka vytiskne:

```
fruits[a] = orange
fruits[b] = banana
fruits[c] = apple
fruits[d] = lemon
```

Vlastnosti třídění lze upravit pomocí volitelného argumentu *`sort_flags`*, detaily viz **sort()**.

Viz také: **asort()**, **arsort()**, **sort()**, **natsort()** a **rsort()**.

> **Poznámka:** Druhý argument byl přidán v PHP 4.

list (unknown)

Přiřadit hodnoty přoměnným jako kdyby byly polem

```
void list(...);
```

Stejně jako **array()**, **list()** vlastně není funkce, ale jazykový konstrukt. Používá se k přiřazení hodnot více proměnným v jedné operaci.

Příklad 1. Ukázka list()

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>

<?php

$result = mysql ($conn, "SELECT id, name, salary FROM employees");
while (list ($id, $name, $salary) = mysql_fetch_row ($result)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php3?id=$id\">$name</a></td>\n".
           "  <td>$salary</td>\n".
           " </tr>\n");
}

?>

</table>
```

Viz také: **each()** a **array()**.

natsort (PHP 4 >= 4.0RC2)

Třídit pole s využitím algoritmu "přirozeného třídění"

```
void natsort (array array)
```

Tato funkce implementuje srovnávací algoritmus který třídí alfanumerické řetězce stejným způsobem jako člověk, toto se popisuje jako "přirozené třídění". Ukázka rozdílu mězi tímto algoritmem a běžnými počítačovými algoritmy pro řazení řetězců (např. **sort()**):

Příklad 1. Ukázka natsort()

```
$array1 = $array2 = array ("img12.png","img10.png","img2.png","img1.png");

sort($array1);
echo "Standardní třídění\n";
print_r($array1);

natsort($array2);
echo "\nPřirozené třídění\n";
print_r($array2);
```

Výše uvedený kód vygeneruje následující výstup:

```
Standardní třídění
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Přirozené třídění
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Více informací viz stránka Martina Poola Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Viz také: **natcasesort()**, **strnatcmp()** a **strnatcasecmp()**.

natcasesort (PHP 4 >= 4.0RC2)

Třídit pole s využitím algoritmu "přirozeného třídění" (case-insensitive)

```
void natcasesort (array array)
```

Tato funkce implementuje srovnávací algoritmus který třídí alfanumerické řetězce stejným způsobem jako člověk, toto se popisuje jako "přirozené třídění".

natcasesort() je case-insensitive verze **natsort()**. Ukázka rozdílu mezi tímto algoritmem a běžným počítačovým tříděním řetězců viz **natsort()**.

Více informací viz stránka Martina Poola Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Viz také: **sort()**, **natsort()**, **strnatcmp()** and **strnatcasecmp()**.

next (PHP 3, PHP 4)

Posunout interní ukazatel pole

```
mixed next (array array)
```

Vrací další prvek pole nebo `false`, pokud prvky došly.

next() se chová jako **current()**, s jedním rozdílem: posouvá interní ukazatel pole o jeden prvek a vrací prvek, na který tento ukazatel ukazuje po posunu. To znamená, že vrací další prvek pole a posouvá interní ukazatel o jeden. Pokud by ukazatel po psunu ukazoval mimo pole, **next()** vrací `false`.

> **Varování**
>
> Pokud toto pole obsahuje prázdné prvky, nebo prvky, jejichž index je 0, tato funkce vrátí `false` i pro tyto prvky. Ke správnému průchodu polem, které může obsahovat prázdné prvky nebo prvky s indexem 0 použijte **each()**.

Viz také: **current()**, **end()**, **prev()** a **reset()**.

pos (PHP 3, PHP 4)

Získat současný prvek pole

```
mixed pos (array array)
```

Toto je alias k **current()**.

Viz také: **end()**, **next()**, **prev()** a **reset()**.

prev (PHP 3, PHP 4)

Rewind interní ukazatel pole

```
mixed prev (array array)
```

Vrací prvek pole před tím prvkem, na který ukazuje interní ukazatel pole, nebo `false`, pokud prvky došly.

> **Varování**
>
> Pokud toto pole obsahuje prázdné prvky, tato funkce vrátí `false` i pro tyto prvky. Ke správnému průchodu polem, které může obsahovat prázdné prvky použijte **each()**.

prev() se chová stejně jako **next()**, ale posouvá interní ukazatel pole o jedno místo zpátky místo dopředu.

Viz také: **current()**, **end()**, **next()** a **reset()**.

range (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Vytvořit pole obsahující rozsah integerů

```
array range (int low, int high)
```

range() vrací pole integerů od *low* po *high* včetně.

Ukázka použití viz **shuffle()**.

reset (PHP 3, PHP 4)

Nastavit interní ukazatel pole na jeho první prvek

```
mixed reset (array array)
```

reset() přetočí interní ukazatel pole *array* na jeho první prvek.

reset() vrací hodnotu prvního prvku pole.

Viz také: **current()**, **each()**, **next()** a **prev()**.

rsort (PHP 3, PHP 4)

Třídit pole sestupně

```
void rsort (array array [, int sort_flags])
```

Tato funkce sestupně třídí pole.

Příklad 1. Ukázka rsort()

```
$fruits = array ("lemon", "orange", "banana", "apple");
rsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key -> $val\n";
}
```

Tato ukázka zobrazí:

```
fruits[0] = orange
fruits[1] = lemon
fruits[2] = banana
fruits[3] = apple
```

Ovoce bylo setříděno podle abecedy sestupně.

Vlastnosti třídění lze upravit pomocí volitelného argumentu *sort_flags*, detaily viz **sort()**.

Viz také: **arsort()**, **asort()**, **ksort()**, **sort()** a **usort()**.

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Zamíchat pole

```
void shuffle (array array)
```

shuffle() zamíchá (náhodně změní pořadí prvků) pole. Musíte inicializovat generátor náhodných čísel pomocí **srand()**.

Příklad 1. Ukázka shuffle()

```
$numbers = range (1,20);
srand ((double)microtime()*1000000);
shuffle ($numbers);
while (list (, $number) = each ($numbers)) {
    echo "$number ";
}
```

Viz také: **arsort()**, **asort()**, **ksort()**, **rsort()**, **sort()** a **usort()**.

sizeof (PHP 3, PHP 4)

Zjistit počet prvků v poli

```
int sizeof (array array)
```

Vrací počet prvků v poli.

Viz také: **count()**.

sort (PHP 3, PHP 4)

Třídit pole

```
void sort (array array [, int sort_flags])
```

sort() třídí pole. Prvky se uspořádají od nejmenšího k největšímu.

Příklad 1. Ukázka sort()

```
<?php

$fruits = array ("lemon", "orange", "banana", "apple");
sort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "fruits[".$key."] = ".$val;
}

?>
```

Tato ukázka zobrazí:

```
fruits[0] = apple
```

```
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

Ovoce bylo seřazeno podle abecedy.

Vlastnosti třídění lze upravit pomocí volitelného argumentu *`sort_flags`*, který může nabývat těchto hodnot:

Typy třídění:

- SORT_REGULAR - normální porovnávání
- SORT_NUMERIC - numerické porovnávání
- SORT_STRING - textové porovnávání

Viz také: **arsort()**, **asort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **usort()**, **array_multisort()** a **uksort()**.

Poznámka: Druhý argument byl přidán v PHP 4.

uasort (PHP 3>= 3.0.4, PHP 4)

Třídit pole pomocí uživatelsky definované porovnávací funkce se zachováním klíčů

```
void uasort (array array, function cmp_function)
```

Tato funkce třídí pole tak, že si indexy uchovávají spojení s hodnotami. To je užitečné hlavně při třídění asociativních polí, kde je důležité pořadí prvků. Srovnávací funkce je uživatelsky definována.

Poznámka: Ukázky uživatelsky definovaných porovnávacích funkcí viz **usort()** a **uksort()**.

Viz také: **usort()**, **uksort()**, **sort()**, **asort()**, **arsort()**, **ksort()** a **rsort()**.

uksort (PHP 3>= 3.0.4, PHP 4)

Třídit pole podle klíčů pomocí uživatelsky definovane porovnávací funkce

```
void uksort (array array, function cmp_function)
```

Tato funkce třídí pole podle klíčů pomocí uživatelsky definované porovnávací funkce. Pokud potřebujete třídit pole podle komplikovanějších kritérií, měli byste použít tuto funkci.

Příklad 1. Ukázka uksort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");

uksort ($a, "cmp");

while (list ($key, $value) = each ($a)) {
```

```
    echo "$key: $value\n";
}
```

Tato ukázka zobrazí:

```
20: twenty
10: ten
4: four
3: three
```

Viz také: **usort()**, **uasort()**, **sort()**, **asort()**, **arsort()**, **ksort()**, **natsort()** a **rsort()**.

usort (PHP 3>= 3.0.3, PHP 4)

Třídit pole podle hodnot pomocí uživatelsky definované porovnávací funkce

```
void usort (array array, string cmp_function)
```

Tato funkce třídí pole podle hodnot pomocí uživatelsky definované porovnávací funkce. Pokud potřebujete třídit pole podle komplikovanějších kritérií, měli byste použít tuto funkci.

Porovnávací funkce musí vrace integer menší než 0, 0, a větší než 0, pokud je první argument menší než, stejný, nebo větší než druhý argument. Pokud jsou dvě porovnávané hodnoty stejné, jejich pořadí v tříděném poli je nedefinováno.

Příklad 1. Ukázka usort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (3, 2, 5, 6, 1);

usort ($a, "cmp");

while (list ($key, $value) = each ($a)) {
    echo "$key: $value\n";
}
```

Tato ukázka zobrazí:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

Poznámka: V tomto jednoduchém případě by pochopitelně bylo vhodnější použít **rsort()**.

Příklad 2. Ukázka usort() s vícerozměrným polem

```
function cmp ($a, $b) {
    return strcmp($a["fruit"],$b["fruit"]);
}

$fruits[0]["fruit"] = "lemons";
$fruits[1]["fruit"] = "apples";
$fruits[2]["fruit"] = "grapes";

usort($fruits, "cmp");

while (list ($key, $value) = each ($fruits)) {
    echo "\$fruits[$key]: " . $value["fruit"] . "\n";
}
```

Při třídění vícerozměrného pole $a a $b obsahují reference na první index pole.

Tato ukázka zobrazí:

```
$fruits[0]: apples
$fruits[1]: grapes
$fruits[2]: lemons
```

> **Varování**
>
> Použitá quicksort funkce v některých C knihovnách (např. na systémech Solaris) může způsobit zhroucení PHP, pokud porovnávací funkce nevrací konsistentní hodnoty.

Viz také: **uasort()**, **uksort()**, **sort()**, **asort()**, **arsort()**, **ksort()**, **natsort()** a **rsort()**.

III. Aspell funkce

aspell() funkce umožňují ověřit hláskování slova a nabídnout možné opravy.

> **Poznámka:** aspell funguje pouze s velmi starými (do přibližně .27.*) verzemi aspell knihovny. Tento modul, ani tyto verze aspell knihovny už nejsou podporovány. Pokud chcete v PHP použít kontrolu hláskování, použijte místo toho pspell. Využívá pspell knihovnu, a pracuje s novějšími verzemi aspellu.

Budete potřebovat aspell knihovnu dostupnou z: http://aspell.sourceforge.net/.

aspell_new (PHP 3>= 3.0.7, PHP 4)

Načíst nový slovník

```
int aspell_new (string master, string personal)
```

Aspell_new() otevře nový slovník, a vrátí identifikátor spojení na tento slovník využitelný s dalšími aspell funkcemi.

Příklad 1. Aspell_new()

```
$aspell_link=aspell_new ("english");
```

aspell_check (PHP 3>= 3.0.7, PHP 4)

Zkontrolovat slovo

```
boolean aspell_check (int dictionary_link, string word)
```

Aspell_check() zkontroluje hláskování slova a vrátí `true`, pokud je hláskování správné, a `false`, pokud není.

Příklad 1. Aspell_check()

```
$aspell_link=aspell_new ("english");
if (aspell_check ($aspell_link, "testt")) {
    echo "Toto je platné hláskování";
} else {
    echo "Pardon, špatné hláskování";
}
```

aspell_check_raw (PHP 3>= 3.0.7, PHP 4)

Zkontrolovat slovo beze změny velikosti písmen nebo pokusů o ořezání

```
boolean aspell_check_raw (int dictionary_link, string word)
```

Aspell_check_raw() zkontroluje hláskování slova beze změn velikosti písmen nebo pokusů ho jakýmkoliv způsobem ořezat, a vrátí `true`, pokud je hláskování správné, a `false`, pokud není.

Příklad 1. Aspell_check_raw()

```
$aspell_link=aspell_new ("english");
if (aspell_check_raw ($aspell_link, "test")) {
    echo "Toto je platné hláskování";
} else {
    echo "Pardon, špatné hláskování";
}
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4)

Nabídnout možné hláskování slova

```
array aspell_suggest (int dictionary_link, string word)
```

Aspell_suggest() vrátí pole možných hláskování daného slova.

Příklad 1. Aspell_suggest()

```
$aspell_link=aspell_new ("english");

if (!aspell_check ($aspell_link, "test")) {
    $suggestions=aspell_suggest ($aspell_link, "test");

    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Možné hláskování: " . $suggestions[$i] . "<br>";
    }
}
```

IV. BCMath funkce pro výpočty s libovolnou přesností

Tyto funkce jsou dostupné pouze pokud bylo PHP zkonfigurováno s `-enable-bcmath`.

bcadd (PHP 3, PHP 4)

Sečíst dvě čísla s libovolnou přesností

```
string bcadd (string left operand, string right operand [, int scale])
```

Přičte *left operand* k *right operand* a vrátí součet v řetězci. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcsub()**.

bccomp (PHP 3, PHP 4)

Porovnat dvě čísla s libovolnou přesností

```
int bccomp (string left operand, string right operand [, int scale])
```

Porovná *left operand* s *right operand* a vrátí výsledek jako integer. Volitelný argument *scale* se používá k určení počtu desetinných míst použitých při porovnání. Návratová hodnota je 0, pokud jsou si oba operandy rovné. Pokud je *left operand* větší než *right operand*, návratová hodnota je +1, a pokud je *left operand* menší než *right operand*, návratová hodnota je -1.

bcdiv (PHP 3, PHP 4)

Dělit dvě čísla s libovolnou přesností

```
string bcdiv (string left operand, string right operand [, int scale])
```

Vydělí argument *left operand* argumentem *right operand* a vrátí výsledek. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcmul()**.

bcmod (PHP 3, PHP 4)

Získat modulus čísla s libovolnou přesností

```
string bcmod (string left operand, string modulus)
```

Vrátí modulus argumentu *left operand* s použitím argumentu *modulus*.

Viz také **bcdiv()**.

bcmul (PHP 3, PHP 4)

Vynásobit dvě čísla s libovolnou přesností

```
string bcmul (string left operand, string right operand [, int scale])
```

Vynásobí argument *left operand* argumentem *right operand* a vrátí výsledek. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcdiv()**.

bcpow (PHP 3, PHP 4)

Umocnit jedno číslo na jiné s libovolnou přesností

```
string bcpow (string x, string y [, int scale])
```

Umocní *x* na *y*. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcsqrt()**.

bcscale (PHP 3, PHP 4)

Nastavit výchozí škálu pro všechny bc math funkce

```
string bcscale (int scale)
```

Tato funkce nastaví výchozí škálu pro všechna následná volání bc math funkcí, ktera neudávají explicitně škálu přesnosti.

bcsqrt (PHP 3, PHP 4)

Získat druhou odmocninu čísla s libovolnou přesností

```
string bcsqrt (string operand, int scale)
```

Vrátí druhou odmocninu argumentu *operand*. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcpow()**.

bcsub (PHP 3, PHP 4)

Odečíst jedno číslo od druhého s libovolnou přesností

```
string bcsub (string left operand, string right operand [, int scale])
```

Odečte argument *right operand* od argumentu *left operand* a vrátí výsledek v řetězci. Volitelný argument *scale* se používá k určení počtu desetinných míst ve výsledku.

Viz také **bcadd()**.

V. Bzip2 Compression Functions

This module uses the functions of the bzip2 (http://sources.redhat.com/bzip2/) library by Julian Seward to transparently read and write bzip2 (.bz2) compressed files.

bzip2 support in PHP is not enabled by default. You will need to use the *-with-bz2[=DIR]* configuration option when compiling php to enable bzip2 support. This module requires bzip2/libbzip2 version >= 1.0.x.

Small code example

This example opens a temporary file and writes a test string to it, then prints out the contents of the file.

Příklad 1. Small bzip2 Example

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Close a bzip2 file pointer

```
int bzclose (int bz)
```

Closes the bzip2 file referenced by the pointer *bz*.

Returns true on success and false on failure.

The file pointer must be valid, and must point to a file successfully opened by **bzopen()**.

See also **bzopen()**.

bzcompress (PHP 4 >= 4.0.4)

Compress a string into bzip2 encoded data

```
string bzcompress (string source [, int blocksize [, int workfactor]])
```

bzcompress() compresses the *source* string and returns it as bzip2 encoded data.

The optional parameter *blocksize* specifies the blocksize used during compression and should be a number from 1 to 9 with 9 giving the best compression, but using more resources to do so. *blocksize* defaults to 4.

The optional parameter *workfactor* controls how the compression phase behaves when presented with worst case, highly repetitive, input data. The value can be between 0 and 250 with 0 being a special case and 30 being the default value. Regardless of the *workfactor*, the generated output is the same.

Příklad 1. bzcompress() Example

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

See also **bzdecompress()**.

bzdecompress (PHP 4 >= 4.0.4)

Decompresses bzip2 encoded data

```
string bzdecompress (string source [, int small])
```

bzdecompress() decompresses the *source* string containing bzip2 encoded data and returns it. If the optional parameter *small* is true, an alternative decompression algorithm will be used which uses less memory (the maximum memory requirement drops to around 2300K) but works at roughly half the speed. See the bzip2 documentation (http://sources.redhat.com/bzip2/) for more information about this feature.

Příklad 1. bzdecompress()

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);
```

```
print( "Compressed String: " );
print( $bzstr );
print( "\n<br>n" );

$str = bzdecompress($bzstr);
print( "Decompressed String: " );
print( $str );
print( "\n<br>n" );
?>
```

See also **bzcompress()**.

bzerrno (PHP 4 >= 4.0.4)

Returns a bzip2 error number

```
int bzerrno (int bz)
```

Returns the error number of any bzip2 error returned by the file pointer *bz*.

See also **bzerror()** and **bzerrstr()**.

bzerror (PHP 4 >= 4.0.4)

Returns the bzip2 error number and error string in an array

```
array bzerror (int bz)
```

Returns the error number and error string, in an associative array, of any bzip2 error returned by the file pointer *bz*.

Příklad 1. bzerror() Example

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

See also **bzerrno()** and **bzerrstr()**.

bzerrstr (PHP 4 >= 4.0.4)

Returns a bzip2 error string

```
string bzerrstr (int bz)
```

Returns the error string of any bzip2 error returned by the file pointer *bz*.

See also **bzerrno()** and **bzerror()**.

bzflush (PHP 4 >= 4.0.4)

Force a write of all buffered data

```
int bzflush (int bz)
```

Forces a write of all buffered bzip2 data for the file pointer *bz*.

Returns true on success, false on failure.

See also **bzread()** and **bzwrite()**.

bzopen (PHP 4 >= 4.0.4)

Open a bzip2 compressed file

```
int bzopen (string filename, string mode)
```

Opens a bzip2 (.bz2) file for reading or writing. *filename* is the name of the file to open. *mode* is similar to the **fopen()** function ('r' for read, 'w' for write, etc.).

If the open fails, the function returns false, otherwise it returns a pointer to the newly opened file.

Příklad 1. bzopen() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$decompressed_file = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "The contents of /tmp/foo.bz2 are: " );
print( "\n<br>n" );
print( $decompressed_file );
?>
```

See also **bzclose()**.

bzread (PHP 4 >= 4.0.4)

Binary safe bzip2 file read

```
string bzread (int bz [, int length])
```

bzread() reads up to *length* bytes from the bzip2 file pointer referenced by *bz*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first. If the optional parameter *length* is not specified, **bzread()** will read 1024 (uncompressed) bytes at a time.

Příklad 1. bzread() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

See also **bzwrite()** and **bzopen()**.

bzwrite (PHP 4 >= 4.0.4)

Binary safe bzip2 file write

```
int bzwrite (int bz, string data [, int length])
```

bzwrite() writes the contents of the string *data* to the bzip2 file stream pointed to by *bz*. If the optional *length* argument is given, writing will stop after length (uncompressed) bytes have been written or the end of string is reached, whichever comes first.

Příklad 1. bzwrite() Example

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

See also **bzread()** and **bzopen()**.

VI. Kalendářové funkce

Kalendářové funkce jsou přístupné pouze pokud máte zkompilovanout calendar extenzi umístěnou v adresáři "dl" or "ext" ve zdrojovém kódu PHP. Před jejím použitím si prosím přečtěte README soubor.

Calendar extenze představuje sadu funkcí určených ke zjednodušení převodů mezi různými kalendáři. Prostředníkem nebo standardem, na kterém je založena je Julian Day Count. To je počet dní začínající daleko před jakýmkoli datem o které by se většina lidí zajímala (někde kolem 4000 př. n. l.). Pokud chcete převádět mezi kalendářovými systémy, musíte nejdřív převést na Julian Day Count, potom na kýžený kalendář. Julian Day Count se velmi liší od Juliánského kaledáře! Více informací o kalendářových systémech viz http://genealogy.org/~scottlee/cal-overview.html. Tyto instrukce obsahují výňatky z této stránky (v uvozovkách).

JDToGregorian (PHP 3, PHP 4)

Převést Julian Day Count na Gregoriánské datum

```
string jdtogregorian (int julianday)
```

Převádí Julian Day Count na řetězec obsahující Gregoriánské datum ve formátu "měsíc/den/rok".

GregorianToJD (PHP 3, PHP 4)

Převést Gregoriánské datum na Julian Day Count

```
int gregoriantojd (int month, int day, int year)
```

Platný rozsah Gregoriánského kalendáře je 4714 př. n. l. až 9999 n. l.

Jakkoli tento software zvládá data až do 4714 př. n. l., takové použití asi nemá smysl. Gregoriánský kalendář byl založen až 15. října 1582 (5. října 1582 podle Juliánského kalendáře). Některé země ho přijaly mnohem později, Řecko až v r. 1923. Většina evropským států před Gregoriánským kalendářem používala Juliánský.

Příklad 1. Kalendářové funkce

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDToJulian (PHP 3, PHP 4)

Převést Julian Day Count na Juliánské datum

```
string jdtojulian (int julianday)
```

Převádí Julian Day Count na řetězec obsahující Juliánské datum ve formátu "měsíc/den/rok".

JulianToJD (PHP 3, PHP 4)

Převést Juliánské datum na Julian Day Count

```
int juliantojd (int month, int day, int year)
```

Platný rozsah pro Juliánský kalendář je 4713 př. n. l. až 9999 n. l.

Jakkoli tento software zvládá data až do 4713 př. n. l., takové použití asi nemá smysl. Tento kalendář byl vytvořen v roce 46 př. n. l., ale detaily se nestabilizovaly nejméně do 8 n. l., a možná až do konce 4. století. Navíc začátek roku se lišil od kultury ke kultuře - ne všechny přijímaly leden jako první měsíc roku.

JDToJewish (PHP 3, PHP 4)

Převést Julian Day Count na idovský kalendář

```
string jdtojewish (int julianday)
```

Převádí Julian Day Count na idovský kalendář.

JewishToJD (PHP 3, PHP 4)

Převést datum podle idovského kalendáře na Julian Day Count

```
int jewishtojd (int month, int day, int year)
```

Platný rozsah Jakkoli tento software zvládá data až do roku 1 (3761 př. n. l.), takové použití asi nemá smysl.

idovský kalendář se používá několik tisíc let, ale zpočátku neexistoval vzorec na určení začátku měsíce. Nový měsíc začínal, když byl poprvé spatřen nový Měsíc.

JDToFrench (PHP 3, PHP 4)

Převést Julian Day Count na Francouzský republikánský kalendář

```
string jdtofrench (int juliandaycount)
```

Převádí Julian Day Count na Francouzský republikánský kalendář.

FrenchToJD (PHP 3, PHP 4)

Převést datum z Francouzského republikánského kalendáře na Julian Day Count

```
int frenchtojd (int month, int day, int year)
```

Převádí datum z Francouzského republikánského kalendáře na Julian Day Count.

Tyto rutiny konvertují pouze data v letech 1 až 14 (Gregoriánská data 22. září 1792 až 22. září 1806). To více než dostatečně pokrývá období, po které se tento kalendář používal.

JDMonthName (PHP 3, PHP 4)

Vrátit název měsíce

```
string jdmonthname (int julianday, int mode)
```

Vrací řetězec obsahující název měsíce. *mode* určuje, na který kalendář se má Julian Day Count konvertovat a jaký typ jména se má vrátit.

Tabulka 1. Kalendářové módy

Mód	Význam
0	Gregoriánský - zkrácený

Mód	Význam
1	Gregoriánský
2	Juliánský - zkrácený
3	Juliánský
4	idovský
5	Francouzský republikánský

JDDayOfWeek (PHP 3, PHP 4)

Vrátit den v týdnu

```
mixed jddayofweek (int julianday, int mode)
```

Vrací den v týdnu. V závislosti na módu vrací řetězec nebo integer.

Tabulka 1. Kalendářové týdenní módy

Mód	Význam
0	Vrací číslo dne jako integer (0=sunday, 1=monday, etc)
1	Vrací řetězec obsahující název dne v týdnu (anglický gregoriánský)
2	Vrací řetězec obsahující zkrácený název dne v týdnu (anglický gregoriánský)

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

Zjistit UNIXový timestamp Velikonoční půlnoci v daném roce

```
int easter_date (int year)
```

Vrací UNIXový timestamp odpovídající Velikonoční půlnoci v daném roce. Default `year` je současný rok.

Varování: Tato funkce vygeneruje varování, pokud je `year` mimo rozsah UNIXových timestampů (tj. před 1970 nebo po 2037).

Příklad 1. Ukázka easter_date()

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

Datum Velikonoc bylo definováno Nicaejským koncilem v r. 325 n. l. jako neděle po prvním úplňku který připadá na nebo po jarní rovnodennosti. Rovnodennost se vždy předpokládá na 21. března, takže se výpočet redukuje na určení data úplňku a data následující neděle. Zde použitý algoritmus byl poprvé použit kolem roku 532 Dionysiem Exiguem. V Juliánském kalendáři (pro léta před 1753) se na sledování fází Měsíce používal jednoduchý devatenáctiletý cyklus. V Gregoriánském kalendáři (pro léta po 1753 - navržen Claviem a Liliem a zaveden papežem Řehořem XIII v říjnu 1582, v Británii a jejích koloniích v září 1752) se přidávají dva faktory, které tento cyklus zpřesňují.

(Kód je založen na C programu od Simona Kershawa, <webmaster@ely.anglican.org>)

Výpočet Velikonoc před rokem 1970 nebo po roce 2037 viz **easter_days()**.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

Get number of days after March 21 on which Easter falls for a given year

```
int easter_days (int year)
```

Vrací počet dní od 21. března do Velikonoc v daném roce. Default *year* je současný rok.

Tato funkce je využitelná místo **easter_date()** na výpočty Velikonoc pro roky které spadají mimo rozsah UNIXových timestampů (tj. před rokem 1970 a po roce 2037).

Příklad 1. Ukázka easter_date()

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

Datum Velikonoc bylo definováno Nicaejským koncilem v r. 325 n. l. jako neděle po prvním úplňku který připadá na nebo po jarní rovnodennosti. Rovnodennost se vždy předpokládá na 21. března, takže se výpočet redukuje na určení data úplňku a data následující neděle. Zde použitý algoritmus byl poprvé použit kolem roku 532 Dionysiem Exiguem. V Juliánském kalendáři (pro léta před 1753) se na sledování fází Měsíce používal jednoduchý devatenáctiletý cyklus. V Gregoriánském kalendáři (pro léta po 1753 - navržen Claviem a Liliem a zaveden papežem Řehořem XIII v říjnu 1582, v Británii a jejích koloniích v září 1752) se přidávají dva faktory, které tento cyklus zpřesňují.

(Kód je založen na C programu od Simona Kershawa, <webmaster@ely.anglican.org>)

Viz také: **easter_date()**.

unixtojd (PHP 4 >= 4.0RC2)

Převést UNIXový timestamp na Julian Day Count

```
int unixtojd ([int timestamp])
```

Vrací Julian Day Count pro UNIXový *timestamp* (sekundy od 1.1.1970), nebo pro aktuální den, pokud není dán *timestamp*.

Viz také: **jdtounix()**.

Poznámka: Tato funkce byla přidána v PHP 4 RC2.

jdtounix (PHP 4 >= 4.0RC2)

Převést Julian Day Count na UNIXový timestamp

```
int jdtounix (int jday)
```

jdtounix() vrací UNIXový timestamp odpovídající Julian Day Countu danému v *jday* nebo `false`, pokud je *jday* mimo UNIXovou epochu (Gregoriánské roky mezi 1970 a 2037, nebo-li 2440588 <= *jday* <= 2465342)

Viz také: **jdtounix()**.

Poznámka: Tato funkce byla přidána v PHP 4 RC2.

VII. CCVS API Funkce

Tyto funkce představují interface k CCVS API, a umožnují tak přímo pracovat s CCVS z vašich PHP skriptů. CCVS je RedHatí (http://www.redhat.com/) řešení "zprostředkovatele" ve zpracování kreditních karet. Umož%nuje vám oslovovat přímo zpracovatele kreditních karet přes váš *nix systém a modem. Pomocí CCVS modulu pro PHP můžete zpracovávat kreditní karty přes CCVS ve vašich PHP skriptech. Následující reference tento proces přiblíží.

Pokud chcete zapnout CCVS podporu v PHP, zjistěte si nejdříve instalační adresář CCVS. Potom budete muset PHP zkonfigurovat s `-with-ccvs`. Pokud toto použijete be udání cesty k vaší instalaci CCVS, PHP se pokusí podívat do defaultní instalační lokace CCVS (/usr/local/ccvs). Pokud je CCVS na nestandardním místě, spustěte configure s: `-with-ccvs=$ccvs_path`, kde $ccvs_path je cesta k vaší instalaci CVS. Pozn.: Podpora CCVS vyžaduje existenci $ccvs_path/lib a $ccvs_path/include, a přítomnost cv_api.h v adresáři include a libccvs.a v adresáři lib.

Dále je potřeba, aby běžel proces ccvsd. Navíc, PHP processy musí běhat pod stejným uživatelem, pod kterým běhá CCVS (např. pokud jste instalovali ccvs jako ’ccvs’, vaše PHP procesy musí také běhat jako ’ccvs’).

Další informace o CCVS jsou na http://www.redhat.com/products/ccvs.

Na této sekci dokumentace se pracuje. Prozatím RedHat udržuje mírně zastaralou, ale stále užitečnou dokumentaci na http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

(unknown)

( )

VIII. Funkce na podporu COM ve Windows

Tyto funkce jsou dostupné pouze ve Windows verzi PHP. Tyto funkce byly přidány v PHP 4.

com_load (PHP 3>= 3.0.3, PHP 4)

Creates a new reference to a COM component

```
string com_load (string module name [, string server name])
```

com_load() vytváří novou COM komponentu a vrací odkaz na ni. Při neúspěchu vrací `false`.

com_invoke (PHP 3>= 3.0.3, PHP 4)

Volá metodu COM komponenty.

```
mixed com_invoke (resource com_object, string function_name [, mixed function
parameters, ...])
```

Com_invoke() volá metodu COM komponenty odkazované *`com_object`*. Pokud dojde k chybě, vrací `false`, při úspěchu vrací návratovou hodnotu metody *`function_name`*.

com_propget (PHP 3>= 3.0.3, PHP 4)

Získává hodnotu vlastnosti COM komponenty

```
mixed com_propget (resource com_object, string property)
```

Tato funkce je alias k **com_get()**.

com_get (PHP 3>= 3.0.3, PHP 4)

Získává hodnotu vlastnosti COM komponenty

```
mixed com_get (resource com_object, string property)
```

Vrací hodnotu *`property`* COM komponenty odkazované *`com_object`*. Při chybě vrací `false`.

com_propput (PHP 3>= 3.0.3, PHP 4)

Přiřazuje hodnotu vlastnosti COM komponenty.

```
void com_propput (resource com_object, string property, mixed value)
```

Tato funkce je alias ke **com_set()**.

com_propset (PHP 3>= 3.0.3, PHP 4)

Přiřazuje hodnotu vlastnosti COM komponenty.

```
void com_propset (resource com_object, string property, mixed value)
```

Tato funkce je alias ke **com_set()**.

com_set (PHP 3>= 3.0.3, PHP 4)

Přiřazuje hodnotu vlastnosti COM komponenty.

```
void com_set (resource com_object, string property, mixed value)
```

Nastavuje hodnotu vlastnosti *property* COM komponent odkazované *com_object*. Vrací true, pokud je *property* nastaveno. Při chybě vrací false.

IX. Funkce pro práci s třídami/objekty

Úvod

About

Tyto funkce vám umožňují získávat informace o třídách a instancích. Můžete zjistit název třídy do které objekt patří nebo jeho proměnné a metody. Pomocí těchto funkcí můžete zjistit nejen příslušnost objektu k třídě, ale i jeho předka (tj. kterou třídu třída tohoto objektu rozšiřuje).

Ukázka použití

V této ukázce nejdříve definujeme základní třídu a rozšíření této třídy. Základní třída popisuje obecnou zeleninu, ať už je jedlá nebo ne a bez ohledu na její barvu. Podtřída `Spenat` přidává metodu na uvaření této zeleniny a další, která zjistí, jestli je vařená.

Příklad 1. classes.inc

```
<?php

// základní třída s členskými proměnnými a metodami
class Zelenina {

    var $jedla;
    var $barva;

    function Zelenina( $jedla, $barva="green" ) {
        $this->jedla = $jedla;
        $this->barva = $barva;
    }

    function je_jedla() {
        return $this->jedla;
    }

    function jaka_barva() {
        return $this->barva;
    }

} // konec tridy zelenina


// rozsiruje zakladni tridu
class Spenat extends Zelenina {

    var $varena = false;

    function Spenat() {
        $this->Zelenina( true, "zelena" );
    }

    function cook_it() {
        $this->varena = true;
    }

    function je_varena() {
        return $this->varena;
    }

} // konec tridy Spenat
```

```
?>
```

Potom z těchto tříd vytvoříme 2 objekty a vytiskneme informace o nich, vč. rodičovských tříd. Také definujeme některé pomocné funkce, především kvůli pohodlnému tisku informací.

Příklad 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// pomocné funkce

function vytiskni_promenne($obj) {
    $pole = get_object_vars($obj);
    while (list($vlastnost, $hodnota) = each($pole))
        echo "\t$vlastnost = $hodnota\n";
}

function vytiskni_metody($obj) {
    $pole = get_class_methods(get_class($obj));
    foreach ($pole as $metoda)
        echo "\tfunction $metoda()\n";
}

function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "Objekt $obj patri do tridy ".get_class($$obj);
        echo " ktera je podtridou $class\n";
    } else {
        echo "Objekt $obj nepatri do podtridy tridy $class\n";
    }
}

// instancujeme 2 objekty

$zeleninka = new Zelenina(true,"modra");
$listnaty = new Spenat();

// vytisknout informace o obou objektech
echo "zeleninka: CLASS ".get_class($zeleninka)."\n";
echo "listnaty: CLASS ".get_class($listnaty);
echo ", PARENT ".get_parent_class($listnaty)."\n";

// vytisknout vlastnosti zeleninky
echo "\nzeleninka: Vlastnosti\n";
print_vars($zeleninka);

// a metody objektu listnaty
echo "\nlistnaty: Metody\n";
print_methods($listnaty);

echo "\nRodic:\n";
class_parentage("listnaty", "Spenat");
class_parentage("listnaty", "Zelenina");
?>
</pre>
```

Je třeba poznamenat, že ve výše uvedené ukázce je objekt `$listnaty` instancí třídy `Spenat`, která je podtřídou třídy `Zelenina`, a poslední část výše uvedeného skriptu tudíž vytiskne:

```
       [...]
Rodic:
Objekt listnaty nepatri do podtridy tridy Spenat
Object listnaty patri do tridy Spenat, ktera je podtridou tridy Zelenina
```

call_user_method (PHP 3>= 3.0.3, PHP 4)

Zavolat uživatelsky definouvanou metodu určitého objektu

```
mixed call_user_method  (string method_name, object obj [, mixed parameter [, mixed
...]])
```

Zavolá metodu *method_name* objektu *obj*. Ukázka využití je níže, kde definujeme třídu, vytvoříme objekt a použijeme **call_user_method()** k nepřímému volání její `print_info` metody.

```
<?php
class Country {
    var $NAME;
    var $TLD;

    function Country($name, $tld) {
        $this->NAME = $name;
        $this->TLD = $tld;
    }

    function print_info($prestr="") {
        echo $prestr."Country: ".$this->NAME."\n";
        echo $prestr."Top Level Domain: ".$this->TLD."\n";
    }
}

$cntry = new Country("Peru","pe");

echo "* Calling the object method directly\n";
$cntry->print_info();

echo "\n* Calling the same method indirectly\n";
call_user_method ("print_info", $cntry, "\t");
?>
```

Viz také **call_user_func()**.

class_exists (PHP 4 >= 4.0b4)

Zjistit, jestli je třída definována

```
bool class_exists (string class_name)
```

Tato funkce vrátí `true`, pokud je třída *class_name* definována, jinak `false`.

get_class (PHP 4 >= 4.0b2)

Vrátit jméno třídy objektu

```
string get_class (object obj)
```

Tato funkce vrací název třídy jíž je objekt *obj* instancí.

Viz také **get_parent_class()**, **is_subclass_of()**

get_class_methods (PHP 4 >= 4.0RC1)

Vrátit pole názvů metod třídy

```
array get_class_methods (string class_name)
```

Tato funkce vrací pole názvů metod definovaných pro třídu specifikovanou argumentem `class_name`.

Viz také **get_class_vars()**, **get_object_vars()**

get_class_vars (PHP 4 >= 4.0RC1)

Vrátit pole defaultních vlastností třídy

```
array get_class_vars (string class_name)
```

Tato funkce vrací pole defaultních vlastností třídy `class_name`.

Viz také **get_class_methods()**, **get_object_vars()**

get_declared_classes (PHP 4 >= 4.0RC2)

Vrátit pole názvů definovaných tříd

```
array get_declared_classes (void)
```

Tato funkce vrací pole názvů tříd definovaných v současném skriptu.

> **Poznámka:** V PHP 4.0.1pl2 jsou na začátku pole vráceny ještě tři další třídy: `stdClass` (definovaná v `Zend/zend.c`), `OverloadedTestClass` (definovaná v `ext/standard/basic_functions.c`) a `Directory` (definovaná v `ext/standard/dir.c`).

get_object_vars (PHP 4 >= 4.0RC1)

Vrátit asociativní pole vlastností objektu

```
array get_object_vars (object obj)
```

Tato funkce vrací asociativní pole definovaných vlastností objektu `obj`. Proměnným deklarovaným v třídě jíž je `obj` instancí, kterým nebyla přiřazena hodnota, nejsou ve vráceném poli obsaženy.

Příklad 1. Použití get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $label;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setLabel($label) {
```

```
        $this->label = $label;
    }

    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}

$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));
// "$label" is declared but not defined
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
// )

$p1->setLabel("point #1");
print_r(get_object_vars($p1));
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
//     [label] => point #1
// )

?>
```

Viz také **get_class_methods()**, **get_class_vars()**

get_parent_class (PHP 4 >= 4.0b2)

Vrátit název rodičovské třídy objektu

```
string get_parent_class (object obj)
```

Tato funkce vrací název rodičovské třídy objektu *obj*.

Viz také **get_class()**, **is_subclass_of()**

is_subclass_of (PHP 4 >= 4.0b4)

Zjistit, jestli objekt patří do podtřídy určité třídy

```
bool is_subclass_of (object obj, string superclass)
```

Tato funkce vrací `true`, pokud je objekt *obj* instancí třídy, která je podtřídou *superclass*, jinak `false`.

Viz také **get_class()**, **get_parent_class()**

method_exists (PHP 4 >= 4.0b2)

Zjistit, jestli má třída určitou metodu

```
bool method_exists (object object, string method_name)
```

Tato funkce vrací `true`, pokud má *`object`* definovánu metodu *`method_name`*, jinak `false`.

X. ClibPDF functions

ClibPDF lets you create PDF documents with PHP. It is available at FastIO (http://www.fastio.com/) but it isn't free software. You should definitely read the licence before you start playing with ClibPDF. If you cannot fullfil the licence agreement consider using pdflib by Thomas Merz, which is also very powerful. ClibPDF functionality and API is similar to Thomas Merz's pdflib but, according to FastIO, ClibPDF is faster and creates smaller documents. This may have changed with the new version 2.0 of pdflib. A simple benchmark (the pdfclock.c example from pdflib 2.0 turned into a php script) actually shows no difference in speed at all. The file size is also similar if compression is turned off. So, try them both and see which one does the job for you.

This documentation should be read alongside the ClibPDF manual since it explains the library in much greater detail.

Many functions in the native ClibPDF and the PHP module, as well as in pdflib, have the same name. All functions except for **cpdf_open()** take the handle for the document as their first parameter.

Currently this handle is not used internally since ClibPDF does not support the creation of several PDF documents at the same time. Actually, you should not even try it, the results are unpredictable. I can't oversee what the consequences in a multi threaded environment are. According to the author of ClibPDF this will change in one of the next releases (current version when this was written is 1.10). If you need this functionality use the pdflib module.

> **Poznámka:** The function **cpdf_set_font()** has changed since PHP 3 to support asian fonts. The encoding parameter is no longer an integer but a string.

One big advantage of ClibPDF over pdflib used to be the possibility to create the pdf document completely in memory without using temporary files. It also provides the ability to pass coordinates in a predefined unit length. This is a handy feature but can be simulated with **pdf_translate()**.

Another nice feature of ClibPDF is the fact that any page can be modified at any time even if a new page has been already opened. The function **cpdf_set_current_page()** allows to leave the current page and presume modifying an other page.

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help you to get started. It creates a document with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is underlined.

Příklad 1. Simple ClibPDF Example

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

The pdflib distribution contains a more complex example which creates a series of pages with an analog clock. Here is that example converted into PHP using the ClibPDF extension:

Příklad 2. pdfclock example from pdflib 2.0 distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;
```

```
$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* minute strokes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* 5 minute strokes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* draw hour hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw minute hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw second hand */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
```

```
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* draw little circle at center */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

cpdf_global_set_document_limits (PHP 4 >= 4.0b4)

Sets document limits for any pdf document

```
void cpdf_global_set_document_limits  (int maxpages, int maxfonts, int maximages, int
maxannotations, int maxobjects)
```

The **cpdf_global_set_document_limits()** function sets several document limits. This function has to be called before **cpdf_open()** to take effect. It sets the limits for any document open afterwards.

See also **cpdf_open()**.

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the creator field in the pdf document

```
void cpdf_set_creator (string creator)
```

The **cpdf_set_creator()** function sets the creator of a pdf document.

See also **cpdf_set_subject()**, **cpdf_set_title()**, **cpdf_set_keywords()**.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the title field of the pdf document

```
void cpdf_set_title (string title)
```

The **cpdf_set_title()** function sets the title of a pdf document.

See also **cpdf_set_subject()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the subject field of the pdf document

```
void cpdf_set_subject (string subject)
```

The **cpdf_set_subject()** function sets the subject of a pdf document.

See also **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the keywords field of the pdf document

```
void cpdf_set_keywords (string keywords)
```

The **cpdf_set_keywords()** function sets the keywords of a pdf document.

See also **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_subject()**.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Opens a new pdf document

```
int cpdf_open (int compression [, string filename])
```

The **cpdf_open()** function opens a new pdf document. The first parameter turns document compression on if it is unequal to 0. The second optional parameter sets the file in which the document is written. If it is omitted the document is created in memory and can either be written into a file with the **cpdf_save_to_file()** or written to standard output with **cpdf_output_buffer()**.

Poznámka: The return value will be needed in further versions of ClibPDF as the first parameter in all other functions which are writing to the pdf document.

The ClibPDF library takes the filename "-" as a synonym for stdout. If PHP is compiled as an apache module this will not work because the way ClibPDF outputs to stdout does not work with apache. You can solve this problem by skipping the filename and using **cpdf_output_buffer()** to output the pdf document.

See also **cpdf_close()**, **cpdf_output_buffer()**.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Closes the pdf document

```
void cpdf_close (int pdf document)
```

The **cpdf_close()** function closes the pdf document. This should be the last function even after **cpdf_finalize()**, **cpdf_output_buffer()** and **cpdf_save_to_file()**.

See also **cpdf_open()**.

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Starts new page

```
void cpdf_page_init (int pdf document, int page number, int orientation, double height,
double width [, double unit])
```

The **cpdf_page_init()** function starts a new page with height *height* and width *width*. The page has number *page number* and orientation *orientation*. *orientation* can be 0 for portrait and 1 for landscape. The last optional parameter *unit* sets the unit for the coordinate system. The value should be the number of postscript points per unit. Since one inch is equal to 72 points, a value of 72 would set the unit to one inch. The default is also 72.

See also **cpdf_set_current_page()**.

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Ends page

```
void cpdf_finalize_page (int pdf document, int page number)
```

The **cpdf_finalize_page()** function ends the page with page number *page number*.

This function is only for saving memory. A finalized page takes less memory but cannot be modified anymore.

See also **cpdf_page_init()**.

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ends document

```
void cpdf_finalize (int pdf document)
```

The **cpdf_finalize()** function ends the document. You still have to call **cpdf_close()**

See also **cpdf_close()**.

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Outputs the pdf document in memory buffer

```
void cpdf_output_buffer (int pdf document)
```

The **cpdf_output_buffer()** function outputs the pdf document to stdout. The document has to be created in memory which is the case if **cpdf_open()** has been called with no filename parameter.

See also **cpdf_open()**.

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Writes the pdf document into a file

```
void cpdf_save_to_file (int pdf document, string filename)
```

The **cpdf_save_to_file()** function outputs the pdf document into a file if it has been created in memory.

This function is not needed if the pdf document has been open by specifying a filename as a parameter of **cpdf_open()**.

See also **cpdf_output_buffer()**, **cpdf_open()**.

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Sets current page

```
void cpdf_set_current_page (int pdf document, int page number)
```

The **cpdf_set_current_page()** function set the page on which all operations are performed. One can switch between pages until a page is finished with **cpdf_finalize_page()**.

See also **cpdf_finalize_page()**.

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Starts text section

```
void cpdf_begin_text (int pdf document)
```

The **cpdf_begin_text()** function starts a text section. It must be ended with **cpdf_end_text()**.

Příklad 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also **cpdf_end_text()**.

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ends text section

```
void cpdf_end_text (int pdf document)
```

The **cpdf_end_text()** function ends a text section which was started with **cpdf_begin_text()**.

Příklad 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also **cpdf_begin_text()**.

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Output text at current position

```
void cpdf_show (int pdf document, string text)
```

The **cpdf_show()** function outputs the string in *text* at the current position.

See also **cpdf_text()**, **cpdf_begin_text()**, **cpdf_end_text()**.

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Output text at position

```
void cpdf_show_xy (int pdf document, string text, double x-coor, double y-coor [, int
mode])
```

The **cpdf_show_xy()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

Poznámka: The function **cpdf_show_xy()** is identical to **cpdf_text()** without the optional parameters.

See also **cpdf_text()**.

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Output text with parameters

```
void cpdf_text (int pdf document, string text, double x-coor, double y-coor [, int mode
[, double orientation [, int alignmode]]])
```

The **cpdf_text()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit. The optional parameter *orientation* is the rotation of the text in degree. The optional parameter *alignmode* determines how the text is aligned.

See the ClibPDF documentation for possible values.

See also **cpdf_show_xy()**.

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Select the current font face and size

```
void cpdf_set_font (int pdf document, string font name, double size, string encoding)
```

The **cpdf_set_font()** function sets the current font face, font size and encoding. Currently only the standard postscript fonts are supported.

The last parameter *encoding* can take the following values: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", and "NULL". "NULL" stands for the font's built-in encoding.

See the ClibPDF Manual for more information, especially how to support asian fonts.

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets distance between text lines

```
void cpdf_set leading (int pdf document, double distance)
```

The **cpdf_set_leading()** function sets the distance between text lines. This will be used if text is output by **cpdf_continue_text()**.

See also **cpdf_continue_text()**.

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Determines how text is rendered

```
void cpdf_set_text_rendering (int pdf document, int mode)
```

The **cpdf_set_text_rendering()** function determines how text is rendered.

The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to clipping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to clipping path, 7=add it to clipping path.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets horizontal scaling of text

```
void cpdf_set_horiz_scaling (int pdf document, double scale)
```

The **cpdf_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the text rise

```
void cpdf_set_text_rise (int pdf document, double value)
```

The **cpdf_set_text_rise()** function sets the text rising to *value* units.

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets the text matrix

```
void cpdf_set_text_matrix (int pdf document, array matrix)
```

The **cpdf_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets text position

```
void cpdf_set_text_pos (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_set_text_pos()** function sets the position of text for the next **cpdf_show()** function call.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_show()**, **cpdf_text()**.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets character spacing

```
void cpdf_set_char_spacing (int pdf document, double space)
```

The **cpdf_set_char_spacing()** function sets the spacing between characters.

See also **cpdf_set_word_spacing()**, **cpdf_set_leading()**.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets spacing between words

```
void cpdf_set_word_spacing (int pdf document, double space)
```

The **cpdf_set_word_spacing()** function sets the spacing between words.

See also **cpdf_set_char_spacing()**, **cpdf_set_leading()**.

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Output text in next line

```
void cpdf_continue_text (int pdf document, string text)
```

The **cpdf_continue_text()** function outputs the string in *text* in the next line.

See also **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_leading()**, **cpdf_set_text_pos()**.

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Returns width of text in current font

```
double cpdf_stringwidth (int pdf document, string text)
```

The **cpdf_stringwidth()** function returns the width of the string in *text*. It requires a font to be set before.

See also **cpdf_set_font()**.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Saves current enviroment

```
void cpdf_save (int pdf document)
```

The **cpdf_save()** function saves the current enviroment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects.

See also **cpdf_restore()**.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Restores formerly saved enviroment

```
void cpdf_restore (int pdf document)
```

The **cpdf_restore()** function restores the enviroment saved with **cpdf_save()**. It works like the postscript command grestore. Very useful if you want to translate or rotate an object without effecting other objects.

Příklad 1. Save/Restore

```
<?php
cpdf_save($pdf);
// do all kinds of rotations, transformations, ...
cpdf_restore($pdf)
?>
```

See also **cpdf_save()**.

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets origin of coordinate system

```
void cpdf_translate (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_translate()** function set the origin of coordinate system to the point (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets scaling

```
void cpdf_scale (int pdf document, double x-scale, double y-scale)
```

The **cpdf_scale()** function set the scaling factor in both directions.

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets rotation

```
void cpdf_rotate (int pdf document, double angle)
```

The **cpdf_rotate()** function set the rotation in degress to *angle*.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets flatness

```
void cpdf_setflat (int pdf document, double value)
```

The **cpdf_setflat()** function set the flatness to a value between 0 and 100.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets linejoin parameter

```
void cpdf_setlinejoin (int pdf document, long value)
```

The **cpdf_setlinejoin()** function set the linejoin parameter between a value of 0 and 2. 0 = miter, 1 = round, 2 = bevel.

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets linecap parameter

```
void cpdf_setlinecap (int pdf document, int value)
```

The **cpdf_setlinecap()** function set the linecap parameter between a value of 0 and 2. 0 = butt end, 1 = round, 2 = projecting square.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets miter limit

```
void cpdf_setmiterlimit (int pdf document, double value)
```

The **cpdf_setmiterlimit()** function set the miter limit to a value greater or equal than 1.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets line width

```
void cpdf_setlinewidth (int pdf document, double width)
```

The **cpdf_setlinewidth()** function set the line width to *width*.

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets dash pattern

```
void cpdf_setdash (int pdf document, double white, double black)
```

The **cpdf_setdash()** function set the dash pattern *white* white units and *black* black units. If both are 0 a solid line is set.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Starts a new path

```
void cpdf_newpath  (int pdf document)
```

The **cpdf_newpath()** starts a new path on the document given by the *pdf document* parameter.

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets current point

```
void cpdf_moveto (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_moveto()** function set the current point to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Sets current point

```
void cpdf_rmoveto (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_rmoveto()** function set the current point relative to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**.

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draws a curve

```
void cpdf_curveto (int pdf document, double x1, double y1, double x2, double y2, double
x3, double y3 [, int mode])
```

The **cpdf_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_rlineto()**, **cpdf_lineto()**.

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draws a line

```
void cpdf_lineto (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Draws a line

```
void cpdf_rlineto (int pdf document, double x-coor, double y-coor [, int mode])
```

The **cpdf_rlineto()** function draws a line from the current point to the relative point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draw a circle

```
void cpdf_circle (int pdf document, double x-coor, double y-coor, double radius [, int mode])
```

The **cpdf_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_arc()**.

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draws an arc

```
void cpdf_arc (int pdf document, double x-coor, double y-coor, double radius, double start, double end [, int mode])
```

The **cpdf_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_circle()**.

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draw a rectangle

```
void cpdf_rect (int pdf document, double x-coor, double y-coor, double width, double height [, int mode])
```

The **cpdf_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *widgth*. This height is set to *height*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Close path

```
void cpdf_closepath (int pdf document)
```

The **cpdf_closepath()** function closes the current path.

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Draw line along path

```
void cpdf_stroke (int pdf document)
```

The **cpdf_stroke()** function draws a line along current path.

See also **cpdf_closepath()**, **cpdf_closepath_stroke()**.

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Close path and draw line along path

```
void cpdf_closepath_stroke (int pdf document)
```

The **cpdf_closepath_stroke()** function is a combination of **cpdf_closepath()** and **cpdf_stroke()**. Then clears the path.

See also **cpdf_closepath()**, **cpdf_stroke()**.

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fill current path

```
void cpdf_fill (int pdf document)
```

The **cpdf_fill()** function fills the interior of the current path with the current fill color.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fill and stroke current path

```
void cpdf_fill_stroke (int pdf document)
```

The **cpdf_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Close, fill and stroke current path

```
void cpdf_closepath_fill_stroke (int pdf document)
```

The **cpdf_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Clips to current path

```
void cpdf_clip (int pdf document)
```

The **cpdf_clip()** function clips all drawing to the current path.

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets filling color to gray value

```
void cpdf_setgray_fill (int pdf document, double value)
```

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.

See also **cpdf_setrgbcolor_fill()**.

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets drawing color to gray value

```
void cpdf_setgray_stroke (int pdf document, double gray value)
```

The **cpdf_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **cpdf_setrgbcolor_stroke()**.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets drawing and filling color to gray value

```
void cpdf_setgray (int pdf document, double gray value)
```

The **cpdf_setgray_stroke()** function sets the current drawing and filling color to the given gray value.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets filling color to rgb color value

```
void cpdf_setrgbcolor_fill (int pdf document, double red value, double green value,
double blue value)
```

The **cpdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets drawing color to rgb color value

```
void cpdf_setrgbcolor_stroke (int pdf document, double red value, double green value,
double blue value)
```

The **cpdf_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sets drawing and filling color to rgb color value

```
void cpdf_setrgbcolor (int pdf document, double red value, double green value, double
blue value)
```

The **cpdf_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Adds bookmark for current page

```
void cpdf_add_outline (int pdf document, string text)
```

The **cpdf_add_outline()** function adds a bookmark with text *text* that points to the current page.

Příklad 1. Adding a page outline

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// some drawing
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Sets duration between pages

```
void cpdf_set_page_animation (int pdf document, int transition, double duration)
```

The **cpdf_set_page_animation()** function set the transition between following pages.

The value of *transition* can be

0 for none,
1 for two lines sweeping across the screen reveal the page,
2 for multiple lines sweeping across the screen reveal the page,
3 for a box reveals the page,
4 for a single line sweeping across the screen reveals the page,
5 for the old page dissolves to reveal the page,
6 for the dissolve effect moves from one screen edge to another,
7 for the old page is simply replaced by the new page (default)

The value of *duration* is the number of seconds between page flipping.

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Opens a JPEG image

```
int cpdf_import_jpeg (int pdf document, string file name, double x-coor, double y-coor,
double angle, double width, double height, double x-scale, double y-scale [, int mode])
```

The **cpdf_import_jpeg()** function opens an image stored in the file with the name *file name*. The format of the image has to be jpeg. The image is placed on the current page at position (*x-coor*, *y-coor*). The image is rotated by *angle* degres.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_place_inline_image()**.

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Places an image on the page

```
void cpdf_place_inline_image (int pdf document, int image, double x-coor, double
y-coor, double angle, double width, double height [, int mode])
```

The **cpdf_place_inline_image()** function places an image created with the php image functions on the page at postion (*x-coor*, *y-coor*). The image can be scaled at the same time.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_import_jpeg()**.

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Adds annotation

```
void cpdf_add_annotation (int pdf document, double llx, double lly, double urx, double
ury, string title, string content [, int mode])
```

The **cpdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

XI. Funkce pro práci s CURL, Client URL Library

PHP podporuje libcurl, knihovnu vytvořenou Danielem Stenbergem, která umožňuje spojení a komunikaci s mnoha různými typy serverů v mnoha různých typech protokolů. libcurl v současné době podporuje http, https, ftp, gopher, telnet, dict, file a ldap protokoly. libcurl také podporuje HTTPS certifikáty, HTTP POST, HTTP PUT, FTP uploady (toto umožňuje i ftp extenze PHP), HTTP formulářové uploady, proxy, cookies a user+password autentikaci.

Pokud chcete používat CURL funkce, musíte nainstalovat CURL (http://curl.haxx.se/). PHP vyžaduje použití CURL 7.0.2-beta nebo vyšší. S verzemi CURL staršími než 7.0.2-beta PHP nebude pracovat.

Dále musíte PHP zkompilovat s `-with-curl[=DIR]`, kde DIR je umístění adresáře obsahujícího lib a include adresáře. V "include" adresáři by měl být adresář pojmenovaný "curl", který by měl obsahovat soubory easy.h and curl.h. V adresáři "lib" by měl být soubor pojmenovaný "libcurl.a".

Tyto funkce byly přidány v PHP 4.0.2.

Pokud máte PHP zkompilované s podporou CURL, můžete začít používat CURL funkce. Základní principem těchto funkcí je, že pomocí **curl_init()** inicializujete CURL session, potom pomocí **curl_exec()** nastavíte hodnoty přenosu a nakonec session zavřete pomocí **curl_close()**. Následuje ukázka, která využíva CURL funkce ke stažení homepage PHP do souboru:

Příklad 1. Použití CURL extenze ke stažení homepage PHP

```
<?php

$ch = curl_init ("http://www.php.net/");
$fp = fopen ("php_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_init (PHP 4 >= 4.0.2)

Inicializovat CURL session

```
int curl_init ([string url])
```

curl_init() inicializuje novou session a vrací CURL handle pro použití s funkcemi **curl_setopt()**, **curl_exec()** a **curl_close()**. Pokud je přítomen volitelný argument *url*, CURLOPT_URL se nastaví na hodnotu tohoto argumentu. Můžete to udělat i ručně, pomocí funkce **curl_setopt()**.

Příklad 1. Inicializace nové CURL session a stažení webové stránky

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

Viz také: **curl_close()**, **curl_setopt()**

curl_setopt (PHP 4 >= 4.0.2)

Nastavit parametr CURL transferu

```
bool curl_setopt (int ch, string option, mixed value)
```

curl_setopt() nastavuje parametry CURL session *ch*. *option* je parametr, který chcete nastavit a *value* je hodnota, na kterou se má *option* nastavit.

Argument *value* by měl u následujících hodnot argumentu *option* obsahovat integer:

- *CURLOPT_INFILESIZE*: Tento parametr by měl u uploadů obsahovat velikost uploadovaného souboru.
- *CURLOPT_VERBOSE*: Pokud chcete, aby CURL podávala zprávy o všem co se děje, nastavte tento parametr na nenulovou hodnotu.
- *CURLOPT_HEADER*: Pokud chcete, aby výstup obsahoval hlavičky, nastavte tento parametr na nenulovou hodnotu.
  - *CURLOPT_NOPROGRESS*: Pokud PHP nemá zobrazit měřidlo postupu CURL transferu, nastavte tento parametr na nenulovou hodnotu.

  Poznámka: PHP tento parametr automaticky nastavuje na nenulovou hodnotu, změna je vhodná pouze pro účely ladění.

- *CURLOPT_NOBODY*: Pokud nechete, aby bylo ve výstupu zahrnuto tělo výstupu, nastavte tento parametr na nenulovou hodnotu.
- *CURLOPT_FAILONERROR*: Pokud má PHP tiše ukončit transfer po přijetí HTTP server kódu většího než 300, nastavte tento parametr na nenulovou hodnotu. Defaultní chování je ignorovat návratový kód a normálně vrátit stránku.
- *CURLOPT_UPLOAD*: Pokud chcete PHP připravit na upload, nastavte tento parametr na nenulovou hodnotu.

- *CURLOPT_POST*: Pokud chcete, aby PHP provedl běžný HTTP POST pořadavek, nastavte tento parametr na nenulovou hodnotu. Jedná se o běžný application/x-www-from-urlencoded POST požadavek, který se většinou používá u HTML formulářů.
- *CURLOPT_FTPLISTONLY*: Pokud chcete, aby PHP vypsalo názvy souborů v FTP adresáři, nastavte tento parametr na nenulovou hodnotu.
- *CURLOPT_FTPAPPEND*: Pokud chcete, aby PHP místo přepsání vzdáleného souboru připojilo upload k jeho obsahu, nastavte tento parametr na nenulovou hodnotu.
- *CURLOPT_NETRC*: Pokud má PHP ve vašem ~./netrc souboru hledat vaše uživatelské jméno a heslo pro server ke kterému se připojujete.
- *CURLOPT_FOLLOWLOCATION*: Pokud má PHP provádět přesměrování u případných "Location: " hlaviček vrácených serverem. (Pozn.: rekurzivní, PHP provede přesměrování pro všechny "Location: " hlavičky, které přijme.)
- *CURLOPT_PUT*: Pokud chcete uploadovat soubor pomocí HTTP metody PUT, nastavte tento parametr na nenulovou hodnotu. Uploadovaný soubor musí být určen parametry CURLOPT_INFILE a CURLOPT_INFILESIZE.
- *CURLOPT_MUTE*: Pokud má být PHP naprosto tiché ohledně CURL funkcí, nastavte tento parametr na nenulovou hodnotu.
- *CURLOPT_TIMEOUT*: Integer určující maximální čas ve vteřinách, který mohou CURL funkce zabrat.
- *CURLOPT_LOW_SPEED_LIMIT*: Integer určující minimální rychlost přenosu v bytech za sekundu. Pokud rychlost přenosu klesne pod tento limit po dobu CURLOPT_LOW_SPEED_TIME sekund, PHP ukončí transfer.
- *CURLOPT_LOW_SPEED_TIME*: Integer určující čas ve vteřinách. Pokud rychlost přenosu klesne na tuto dobu pod CURLOPT_LOW_SPEED_LIMIT, PHP zruší transfer.
- *CURLOPT_RESUME_FROM*: Integer určující offset v bytech, na kterém má transfer začít.
- *CURLOPT_SSLVERSION*: Integer určující, jaká verze SSL (2 nebo 3) se má použít. Defaultně se PHP pokusí určit verzi samo, ale v některých případech je nutno verzi určit manuálně.
- *CURLOPT_TIMECONDITION*: Definující chování CURLOPT_TIMEVALUE. Tento parametr může nabýt buď hodnoty TIMECOND_IFMODSINCE nebo TIMECOND_ISUNMODSINCE. Funguje pouze u HTTP přenosů.
- *CURLOPT_TIMEVALUE*: Integer určující počet vteřin od 1. ledna 1970. Tento čas se použije podle intervalu CURLOPT_TIMEVALUE, default je použití TIMECOND_IFMODSINCE.

Argument *value* by měl u následujících hodnot argumentu *option* obsahovat řetězec:

- *CURLOPT_URL*: Toto je URL, kterou má PHP stáhnout. Tento parametr můžete také nastavit při inicializaci CURL session pomocí funkce **curl_init()**.
- *CURLOPT_USERPWD*: Řetězec ve tvaru [username]:[password] pro použití při spojení.
- *CURLOPT_PROXYUSERPWD*: Řetězec ve tvaru [username]:[password] pro použití při spojení s HTTP proxy.
- *CURLOPT_RANGE*: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, seperated with commas as in X-Y,N-M.
- *CURLOPT_POSTFIELDS*: Řetězec obsahující kompletní data, která se mají odeslat v HTTP POST požadavku.
- *CURLOPT_REFERER*: Řetězec obsahující "referer" hlavičku pro použití v HTTP požadavku.
- *CURLOPT_USERAGENT*: Řetězec obsahující "user-agent" hlavičku pro použití v HTTP požadavku.
- *CURLOPT_FTPPORT*: Řetězec, na jehož základě se získá IP adresa pro FTP "POST" instrukci. POST instrukce říká serveru, aby se připojil na danou IP adresu. Tento řetězec může obsahovat IP adresu, hostname, a network interface name (under UNIX) nebo '-' (použije se defaultní IP adresa systému).
- *CURLOPT_COOKIE*: Řetězec obsahující cookie, který se má poslat v HTTP hlavičce tohoto přenosu.
- *CURLOPT_SSLCERT*: Řetězec obsahující název souboru PEM certifikátu.
- *CURLOPT_SSLCERTPASSWD*: Řetězec obsahující heslo vyžadované pro použití CURLOPT_SSLCERT certifikátu.

- *CURLOPT_COOKIEFILE*: Řetězec obsahující název souboru obsahujícího cookie data. Cookie soubor může být buď v Netscape formátu nebo obsahovat HTTP hlavičky.
    - *CURLOPT_CUSTOMREQUEST*: Řetězec, který se má v HTTP požadavku použít místo GET nebo HEAD. Toto je užitečné při DELETE či jiných, obskurnějších HTTP požadavcích.

Poznámka: Používejte pouze v případě, že váš server tento příkaz podporuje.

Následující parametry očekávají deskriptor vrácený funkcí **fopen()**:

- *CURLOPT_FILE*: Soubor, do kterého se má umístit výstup CURL transferu. Default je STDOUT.
- *CURLOPT_INFILE*: Soubor, který obsahuje vstup CURL transferu.
- *CURLOPT_WRITEHEADER*: Soubor, do kterého se mají zapsat hlavičky výstupu.
- *CURLOPT_STDERR*: Soubor, do kterého se mají zapisovat chyby místo na STDERR.

curl_exec (PHP 4 >= 4.0.2)

Provést CURL session

```
bool curl_exec (int ch)
```

Tuto funkci byste měli zavolat po inicializaci CURL session a nastavení všech jejích parametrů. Jejím účelem je provést předdefinovanou CURL session (určenou argumentem *ch*).

curl_close (PHP 4 >= 4.0.2)

Zavřít CURL session

```
void curl_close  (int ch)
```

Tato funkce zavře CURL session a uvolní všechny zdroje. CURL handle, *ch*, se také smaže.

curl_version (PHP 4 >= 4.0.2)

Vrátit verzi CURL

```
string curl_version (void);
```

curl_version() vrací řetězec obsahující použitou verzi CURL.

XII. Cybercash platební funkce

Tyto funkce jsou dostupné pouze pokud bylo PHP zkompilováno s `-with-cybercash=[DIR]`. Tyto funkce byly přidány v PHP 4.

cybercash_encr (PHP 4 >= 4.0b4)

???

```
array cybercash_encr (string wmk, string sk, string inbuff)
```

Tato funkce vrací asociativní pole s elementem "errcode", a pokud je "errcode" `false`, "outbuff" (string), "outLth" (long) and "macbuff" (string).

cybercash_decr (PHP 4 >= 4.0b4)

???

```
array cybercash_decr (string wmk, string sk, string inbuff)
```

tato funkce vrací asociativní pole s elementem "errcode", a pokud je "errcode" `false`, "outbuff" (string), "outLth" (long) a "macbuff" (string).

cybercash_base64_encode (PHP 4 >= 4.0b4)

???

```
string cybercash_base64_encode  (string inbuff)
```

cybercash_base64_decode (PHP 4 >= 4.0b4)

```
string cybercash_base64_decode  (string inbuff)
```

XIII. Character type functions

These functions check whether a character or string falls into a certain character class according to the i current locale.

When called with an integer argument theese functions behave exactly like their C counterparts.

When called with a string argument they will check every character in the string and will only return true if every character in the string matches the requested criteria.

Passing anything else but a string or integer will return false immediately.

> **Varování**
>
> These functions are new as of PHP 4.0.4 and might change their name in the near future. Suggestions are to change them to **ctype_issomething()** instead of **ctype_somthing()** or even to make them part of `ext/standard` and use their original C-names, although this would possibly lead to further confusion regarding the **isset()** vs. **is_sometype()** problem.

ctype_alnum (PHP 4 >= 4.0.4)

Check for alphanumeric character(s)

```
bool ctype_alnum (string c)
```

See also **setlocale()**.

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

```
bool ctype_alpha (string c)
```

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

```
bool ctype_cntrl (string c)
```

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

```
bool ctype_digit (string c)
```

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

```
bool ctype_lower (string c)
```

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

```
bool ctype_graph (string c)
```

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

```
bool ctype_print (string c)
```

ctype_punct (PHP 4 >= 4.0.4)

Check for any printable character which is not whitespace or an alphanumeric character

```
bool ctype_punct (string c)
```

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

```
bool ctype_space (string c)
```

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

```
bool ctype_upper (string c)
```

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

```
bool ctype_xdigit (string c)
```

XIV. Database (dbm-style) abstraction layer functions

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a subset of features modern databases such as Sleepycat Software's DB2 (http://www.sleepycat.com/) support. (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

The behaviour of various aspects depend on the implementation of the underlying database. Functions such as **dba_optimize()** and **dba_sync()** will do what they promise for one database and will do nothing for others.

To add support for any of the following handlers, add the specified –with configure switch to your PHP configure line:

- Dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format. (–with-dbm)
- Ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated). (–with-ndbm)
- Gdbm is the GNU database manager (ftp://ftp.gnu.org/pub/gnu/gdbm/). (–with-gdbm)
- DB2 is Sleepycat Software's DB2 (http://www.sleepycat.com/). It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications." (–with-db2)
- DB3 is Sleepycat Software's DB3 (http://www.sleepycat.com/). (–with-db3)
- Cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found here (http://pobox.com/~djb/cdb.html). Since it is constant, we support only reading operations. (–with-cdb)

Příklad 1. DBA example

```
<?php

$id = dba_open ("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. It inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to **dba_open()** or **dba_popen()**.

You can access all entries of a database in a linear way by using the **dba_firstkey()** and **dba_nextkey()** functions. You

may not change the database while traversing it.

Příklad 2. Traversing a database

```
<?php

# ...open database...

$key = dba_firstkey ($id);

while ($key != false) {
    if (...) { # remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Close database

```
void dba_close (int handle)
```

Dba_close() closes the established database and frees all resources specified by *handle*.
handle is a database handle returned by **dba_open()**.
Dba_close() does not return any value.
See also: **dba_open()** and **dba_popen()**

dba_delete (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Delete entry specified by key

```
string dba_delete (string key, int handle)
```

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.
key is the key of the entry which is deleted.
handle is a database handle returned by **dba_open()**.
dba_delete() returns true or false, if the entry is deleted or not deleted, respectively.
See also: **dba_exists()**, **dba_fetch()**, **dba_insert()**, and **dba_replace()**.

dba_exists (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Check whether key exists

```
bool dba_exists (string key, int handle)
```

Dba_exists() checks whether the specified *key* exists in the database specified by *handle*.
Key is the key the check is performed for.
Handle is a database handle returned by **dba_open()**.
Dba_exists() returns true or false, if the key is found or not found, respectively.
See also: **dba_fetch()**, **dba_delete()**, **dba_insert()**, and **dba_replace()**.

dba_fetch (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Fetch data specified by key

```
string dba_fetch (string key, int handle)
```

Dba_fetch() fetches the data specified by *key* from the database specified with *handle*.
Key is the key the data is specified by.
Handle is a database handle returned by **dba_open()**.
Dba_fetch() returns the associated string or false, if the key/data pair is found or not found, respectively.
See also: **dba_exists()**, **dba_delete()**, **dba_insert()**, and **dba_replace()**.

dba_firstkey (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Fetch first key

```
string dba_firstkey (int handle)
```

Dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

Handle is a database handle returned by **dba_open()**.

Dba_firstkey() returns the key or false depending on whether it succeeds or fails, respectively.

See also: **Dba_nextkey()**

dba_insert (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Insert entry

```
bool dba_insert (string key, string value, int handle)
```

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_insert() returns true or false, depending on whether it succeeds of fails, respectively.

See also: **dba_exists() dba_delete() dba_fetch() dba_replace()**

dba_nextkey (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Fetch next key

```
string dba_nextkey (int handle)
```

dba_nextkey() returns the next key of the database specified by *handle* and increments the internal key pointer.

handle is a database handle returned by **dba_open()**.

dba_nextkey() returns the key or false depending on whether it succeeds or fails, respectively.

See also: **dba_firstkey()**

dba_popen (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Open database persistently

```
int dba_popen (string path, string mode, string handler [, ...])
```

dba_popen() establishes a persistent database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handler id or false, in the case the open is successful or fails, respectively.

See also: **dba_open() dba_close()**

dba_open (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Open database

```
int dba_open (string path, string mode, string handler [, ...])
```

dba_open() establishes a database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handler id or false, in the case the open is successful or fails, respectively.

See also: **dba_popen() dba_close()**

dba_optimize (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Optimize database

```
bool dba_optimize (int handle)
```

dba_optimize() optimizes the underlying database specified by *handle*.

handle is a database handle returned by **dba_open()**.

dba_optimize() returns true or false, if the optimization succeeds or fails, respectively.

See also: **dba_sync()**

dba_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Replace or insert entry

```
bool dba_replace (string key, string value, int handle)
```

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_replace() returns true or false, depending on whether it succeeds of fails, respectively.

See also: **dba_exists()**, **dba_delete()**, **dba_fetch()**, and **dba_insert()**.

dba_sync (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Synchronize database

```
bool dba_sync (int handle)
```

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by **dba_open()**.

dba_sync() returns true or false, if the synchronization succeeds or fails, respectively.

See also: **dba_optimize()**

XV. Date and Time functions

checkdate (PHP 3, PHP 4)

Validate a gregorian date/time

```
int checkdate (int month, int day, int year)
```

Returns true if the date given is valid; otherwise returns false. Checks the validity of the date formed by the arguments. A date is considered valid if:

- year is between 1 and 32767 inclusive
- month is between 1 and 12 inclusive
- *Day* is within the allowed number of days for the given *month*. Leap *year*s are taken into consideration.

date (PHP 3, PHP 4)

Format a local time/date

```
string date (string format [, int timestamp])
```

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given.

The following characters are recognized in the format string:

- a - "am" or "pm"
- A - "AM" or "PM"
- B - Swatch Internet time
- d - day of the month, 2 digits with leading zeros; i.e. "01" to "31"
- D - day of the week, textual, 3 letters; i.e. "Fri"
- F - month, textual, long; i.e. "January"
- g - hour, 12-hour format without leading zeros; i.e. "1" to "12"
- G - hour, 24-hour format without leading zeros; i.e. "0" to "23"
- h - hour, 12-hour format; i.e. "01" to "12"
- H - hour, 24-hour format; i.e. "00" to "23"
- i - minutes; i.e. "00" to "59"
- I (capital i) - "1" if Daylight Savings Time, "0" otherwise.
- j - day of the month without leading zeros; i.e. "1" to "31"
- l (lowercase 'L') - day of the week, textual, long; i.e. "Friday"
- L - boolean for whether it is a leap year; i.e. "0" or "1"
- m - month; i.e. "01" to "12"
- M - month, textual, 3 letters; i.e. "Jan"
- n - month without leading zeros; i.e. "1" to "12"
- r - RFC 822 formatted date; i.e. "Thu, 21 Dec 2000 16:01:07 +0200" (added in PHP 4.0.4)
- s - seconds; i.e. "00" to "59"
- S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd"
- t - number of days in the given month; i.e. "28" to "31"

- T - Timezone setting of this machine; i.e. "MDT"
- U - seconds since the epoch
- w - day of the week, numeric, i.e. "0" (Sunday) to "6" (Saturday)
- Y - year, 4 digits; i.e. "1999"
- y - year, 2 digits; i.e. "99"
- z - day of the year; i.e. "0" to "365"
- Z - timezone offset in seconds (i.e. "-43200" to "43200"). The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.

Unrecognized characters in the format string will be printed as-is. The "Z" format will always return "0" when using **gmdate()**.

Příklad 1. Date() example

```
print (date ("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000)));
```

It is possible to use **date()** and **mktime()** together to find dates in the future or the past.

Příklad 2. Date() and mktime() example

```
$tomorrow  = mktime (0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime (0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime (0,0,0,date("m"),  date("d"),  date("Y")+1);
```

Some examples of **date()** formatting. Note that you should escape any other characters, as any which currently have a special meaning will produce undesirable results, and other characters may be assigned meaning in future PHP versions. When escaping, bu sure to use single quotes to prevent characters like \n from become newlines.

Příklad 3. Date() Formatting

```
/* Today is March 10th, 2001, 5:16:18 pm */
$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, g, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-
01, 1631 1618 6 Fripm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
```

To format dates in other languages, you should use the **setlocale()** and **strftime()** functions.

See also **gmdate()** and **mktime()**.

getdate (PHP 3, PHP 4)

Get date/time information

```
array getdate ([int timestamp])
```

Returns an associative array containing the date information of the *timestamp*, or the current local time if no timestamp is given, as the following array elements:

- "seconds" - seconds
- "minutes" - minutes
- "hours" - hours
- "mday" - day of the month
- "wday" - day of the week, numeric
- "mon" - month, numeric
- "year" - year, numeric
- "yday" - day of the year, numeric; i.e. "299"
- "weekday" - day of the week, textual, full; i.e. "Friday"
- "month" - month, textual, full; i.e. "January"

Příklad 1. getdate() example

```
$today = getdate();
$month = $today[month];
$mday = $today[mday];
$year = $today[year];
echo "$month $mday, $year";
```

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Get current time

```
array gettimeofday (void)
```

This is an interface to gettimeofday(2). It returns an associative array containing the data returned from the system call.

- "sec" - seconds
- "usec" - microseconds
- "minuteswest" - minutes west of Greenwich
- "dsttime" - type of dst correction

gmdate (PHP 3, PHP 4)

Format a GMT/CUT date/time

```
string gmdate (string format [, int timestamp])
```

Identical to the **date()** function except that the time returned is Greenwich Mean Time (GMT). For example, when run in Finland (GMT +0200), the first line below prints "Jan 01 1998 00:00:00", while the second prints "Dec 31 1997 22:00:00".

Příklad 1. Gmdate() example

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
```

```
echo gmdate ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
```

See also **date()**, **mktime()**, and **gmmktime()**.

gmmktime (PHP 3, PHP 4)

Get UNIX timestamp for a GMT date

```
int gmmktime (int hour, int minute, int second, int month, int day, int year [, int
is_dst])
```

Identical to **mktime()** except the passed parameters represents a GMT date.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Format a GMT/CUT time/date according to locale settings

```
string gmstrftime (string format [, int timestamp])
```

Behaves the same as **strftime()** except that the time returned is Greenwich Mean Time (GMT). For example, when run in Eastern Standard Time (GMT -0500), the first line below prints "Dec 31 1998 20:00:00", while the second prints "Jan 01 1999 01:00:00".

Příklad 1. Gmstrftime() example

```
setlocale ('LC_TIME', 'en_US');
echo strftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
```

See also **strftime()**.

localtime (PHP 4 >= 4.0RC2)

Get the local time

```
array localtime ([int timestamp [, bool is_associative]])
```

The **localtime()** function returns an array identical to that of the structure returned by the C function call. The first argument to **localtime()** is the timestamp, if this is not given the current time is used. The second argument to the **localtime()** is the *is_associative*, if this is set to 0 or not supplied than the array is returned as a regular, numerically indexed array. If the argument is set to 1 then **localtime()** is an associative array containing all the different elements of the structure returned by the C function call to localtime. The names of the different keys of the associative array are as follows:

- "tm_sec" - seconds
- "tm_min" - minutes
- "tm_hour" - hour
- "tm_mday" - day of the month
- "tm_mon" - month of the year

- "tm_year" - Years since 1900
- "tm_wday" - Day of the week
- "tm_yday" - Day of the year
- "tm_isdst" - Is daylight savings time in effect

microtime (PHP 3, PHP 4)

Return current UNIX timestamp with microseconds

```
string microtime(void);
```

Returns the string "msec sec" where sec is the current time measured in the number of seconds since the Unix Epoch (0:00:00 January 1, 1970 GMT), and msec is the microseconds part. This function is only available on operating systems that support the gettimeofday() system call.

Both portions of the string are returned in units of seconds.

Příklad 1. microtime() example

```
function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
    }

$time_start = getmicrotime();

for ($i=0; $i < 1000; $i++){
    //do nothing, 1000 times
    }

$time_end = getmicrotime();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds";
```

See also **time()**.

mktime (PHP 3, PHP 4)

Get UNIX timestamp for a date

```
int mktime (int hour, int minute, int second, int month, int day, int year [, int
is_dst])
```

Warning: Note the strange order of arguments, which differs from the order of arguments in a regular UNIX mktime() call and which does not lend itself well to leaving out parameters from right to left (see below). It is a common error to mix these values up in a script.

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1 1970) and the time specified.

Arguments may be left out in order from right to left; any arguments thus omitted will be set to the current value according to the local date and time.

Is_dst can be set to 1 if the time is during daylight savings time, 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not.

> **Poznámka:** *Is_dst* was added in 3.0.10.

Mktime() is useful for doing date arithmetic and validation, as it will automatically calculate the correct value for out-of-range input. For example, each of the following lines produces the string "Jan-01-1998".

Příklad 1. Mktime() example

```
echo date ("M-d-Y", mktime (0,0,0,12,32,1997));
echo date ("M-d-Y", mktime (0,0,0,13,1,1997));
echo date ("M-d-Y", mktime (0,0,0,1,1,1998));
echo date ("M-d-Y", mktime (0,0,0,1,1,98));
```

Year may be a two or four digit value, with values between 0-69 mapping to 2000-2069 and 70-99 to 1970-1999 (on systems where time_t is a 32bit signed integer, as most common today, the valid range for *year* is somewhere between 1902 and 2037).

The last day of any given month can be expressed as the "0" day of the next month, not the -1 day. Both of the following examples will produce the string "The last day in Feb 2000 is: 29".

Příklad 2. Last day of next month

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);

$lastday = mktime (0,0,0,4,-31,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);
```

Date with year, month and day equal to zero is considered illegal (otherwise it what be regarded as 30.11.1999, which would be strange behaviour).

See also **date()** and **time()**.

strftime (PHP 3, PHP 4)

Format a local time/date according to locale settings

```
string strftime (string format [, int timestamp])
```

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with **setlocale()**.

The following conversion specifiers are recognized in the format string:

- %a - abbreviated weekday name according to the current locale
- %A - full weekday name according to the current locale
- %b - abbreviated month name according to the current locale
- %B - full month name according to the current locale
- %c - preferred date and time representation for the current locale
- %C - century number (the year divided by 100 and truncated to an integer, range 00 to 99)
- %d - day of the month as a decimal number (range 01 to 31)
- %D - same as %m/%d/%y

- %e - day of the month as a decimal number, a single digit is preceded by a space (range ’ 1’ to ’31’)
- %h - same as %b
- %H - hour as a decimal number using a 24-hour clock (range 00 to 23)
- %I - hour as a decimal number using a 12-hour clock (range 01 to 12)
- %j - day of the year as a decimal number (range 001 to 366)
- %m - month as a decimal number (range 01 to 12)
- %M - minute as a decimal number
- %n - newline character
- %p - either ‘am’ or ‘pm’ according to the given time value, or the corresponding strings for the current locale
- %r - time in a.m. and p.m. notation
- %R - time in 24 hour notation
- %S - second as a decimal number
- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a decimal number [1,7], with 1 representing Monday
- %U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week.
- %W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday being 0
- %x - preferred date representation for the current locale without the time
- %X - preferred time representation for the current locale without the date
- %y - year as a decimal number without a century (range 00 to 99)
- %Y - year as a decimal number including the century
- %Z - time zone or name or abbreviation
- %% - a literal ‘%’ character

> **Poznámka:** Not all conversion specifiers may be supported by your C library, in which case they will not be supported by PHP’s **strftime()**.

Příklad 1. Strftime() example

```
setlocale ("LC_TIME", "C");
print (strftime ("%A in Finnish is "));
setlocale ("LC_TIME", "fi_FI");
print (strftime ("%A, in French "));
setlocale ("LC_TIME", "fr_CA");
print (strftime ("%A and in German "));
setlocale ("LC_TIME", "de_DE");
print (strftime ("%A.\n"));
```

This example works if you have the respective locales installed in your system.

See also **setlocale()** and **mktime()** and the Open Group specification of **strftime()** (http://www.opengroup.org/onlinepubs/7908799/xsh/strftime.html).

time (PHP 3, PHP 4)

Return current UNIX timestamp

```
int time(void);
```

Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

See also **date()**.

strtotime (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Parse about any english textual datetime description into a UNIX timestamp

```
int strtotime (string time [, int now])
```

The function expects to be given a string containing an english date format and will try to parse that format into a UNIX timestamp.

Příklad 1. Strtotime() examples

```
echo strtotime ("now") . "\n";
echo strtotime ("10 September 2000") . "\n";
echo strtotime ("+1 day") . "\n";
echo strtotime ("+1 week") . "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds") . "\n";
```

XVI. dBase functions

These functions allow you to access records stored in dBase-format (dbf) databases.

There is no support for indexes or memo fields. There is no support for locking, too. Two concurrent webserver processes modifying the same dBase file will very likely ruin your database.

Unlike SQL databases, dBase "databases" cannot change the database definition afterwards. Once the file is created, the database definition is fixed. There are no indexes that speed searching or otherwise organize your data. dBase files are simple sequential files of fixed length records. Records are appended to the end of the file and delete records are kept until you call **dbase_pack()()**.

We recommend that you do not use dBase files as your production database. Choose any real SQL server instead; MySQL or Postgres are common choices with PHP. dBase support is here to allow you to import and export data to and from your web database, since the file format is commonly understood with Windows spreadsheets and organizers. Import and export of data is about all that dBase support is good for.

dbase_create (PHP 3, PHP 4)

Creates a dBase database

```
int dbase_create (string filename, array fields)
```

The *fields* parameter is an array of arrays, each array describing the format of one field in the database. Each field consists of a name, a character indicating the field type, a length, and a precision.

The types of fields available are:

L

Boolean. These do not have a length or precision.

M

Memo. (Note that these aren't supported by PHP.) These do not have a length or precision.

D

Date (stored as YYYYMMDD). These do not have a length or precision.

N

Number. These have both a length and a precision (the number of digits after the decimal point).

C

String.

If the database is successfully created, a dbase_identifier is returned, otherwise false is returned.

Příklad 1. Creating a dBase database file

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_open (PHP 3, PHP 4)

Opens a dBase database

```
int dbase_open (string filename, int flags)
```

The flags correspond to those for the open() system call. (Typically 0 means read-only, 1 means write-only, and 2 means read and write.)

Returns a dbase_identifier for the opened database, or false if the database couldn't be opened.

dbase_close (PHP 3, PHP 4)

Close a dBase database

```
bool dbase_close (int dbase_identifier)
```

Closes the database associated with *dbase_identifier*.

dbase_pack (PHP 3, PHP 4)

Packs a dBase database

```
bool dbase_pack (int dbase_identifier)
```

Packs the specified database (permanently deleting all records marked for deletion using **dbase_delete_record()**.

dbase_add_record (PHP 3, PHP 4)

Add a record to a dBase database

```
bool dbase_add_record (int dbase_identifier, array record)
```

Adds the data in the *record* to the database. If the number of items in the supplied record isn't equal to the number of fields in the database, the operation will fail and false will be returned.

dbase_replace_record (PHP 3>= 3.0.11, PHP 4)

Replace a record in a dBase database

```
bool dbase_replace_record (int dbase_identifier, array record, int dbase_record_number)
```

Replaces the data associated with the record *record_number* with the data in the *record* in the database. If the number of items in the supplied record is not equal to the number of fields in the database, the operation will fail and false will be returned.

dbase_record_number is an integer which spans from 1 to the number of records in the database (as returned by **dbase_numrecords()**).

dbase_delete_record (PHP 3, PHP 4)

Deletes a record from a dBase database

```
bool dbase_delete_record (int dbase_identifier, int record)
```

Marks *record* to be deleted from the database. To actually remove the record from the database, you must also call **dbase_pack()**.

dbase_get_record (PHP 3, PHP 4)

Gets a record from a dBase database

```
array dbase_get_record (int dbase_identifier, int record)
```

Returns the data from *record* in an array. The array is indexed starting at 0, and includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see **dbase_delete_record()**.

Each field is converted to the appropriate PHP type. (Dates are left as strings.)

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4)

Gets a record from a dBase database as an associative array

```
array dbase_get_record_with_names  (int dbase_identifier, int record)
```

Returns the data from *record* in an associative array. The array also includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see **dbase_delete_record()**.

Each field is converted to the appropriate PHP type. (Dates are left as strings.)

dbase_numfields (PHP 3, PHP 4)

Find out how many fields are in a dBase database

```
int dbase_numfields (int dbase_identifier)
```

Returns the number of fields (columns) in the specified database. Field numbers are between 0 and dbase_numfields($db)-1, while record numbers are between 1 and dbase_numrecords($db).

Příklad 1. Using dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords (PHP 3, PHP 4)

Find out how many records are in a dBase database

```
int dbase_numrecords (int dbase_identifier)
```

Returns the number of records (rows) in the specified database. Record numbers are between 1 and dbase_numrecords($db), while field numbers are between 0 and dbase_numfields($db)-1.

XVII. DBM Funkce

Tyto funkce vám umožňují ukládat záznamy do databází typu dbm. Tento typ databází (podporovaný Berkeley DB, GDBM, některými systémovými knihovnami, a také vestavěnou flatfile knihovnou) ukládá klíč/hodnota páry (oproti plnohodnotným relačním databázím).

Příklad 1. Ukázka DBM

```
$dbm = dbmopen ("lastseen", "w");
if (dbmexists ($dbm, $userid)) {
    $last_seen = dbmfetch ($dbm, $userid);
} else {
    dbminsert ($dbm, $userid, time());
}
do_stuff();
dbmreplace ($dbm, $userid, time());
dbmclose ($dbm);
```

dbmopen (PHP 3, PHP 4)

Otevřít DBM databázi

```
int dbmopen (string filename, string flags)
```

První argument je název DBM souboru (plná cesta), který se má otevřít, druhý argument je jedno z "r" (pouze pro čtení), "n" (nový, implikuje čtení/zápis, a nejspíš smaže existující databázi stejného jména), "c" (vytvořit, implikuje čtení/zápis, a nesmaže existující databázi stejného jména) nebo "w" (čtení/zápis).

Při úspěchu vrací identifikátor, který se předává dalším DBM funkcím success, jinak `false`.

Pokud používáte NDBM, NDBM ve skutečnosti vytváří soubory soubor.dir a soubor.pag. GDBM používá pouze jeden soubr, stejně jako interní flatfile podpora, a Berkeley DB vytváří soubor `filename.db`. Pozn.: Vedle případného zamykání souborů vlastní DBM knihovnou provádí PHP svoje vlastní zamykání souborů. PHP nemaže `.lck` soubory, které vytváří. Používá tyto soubory jako pevné inodes, na kterých provádí zamykání souborů. Více informací o DBM souborech viz vaše Unixové man stránky, nebo si stáhněte GNU GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/).

dbmclose (PHP 3, PHP 4)

Zavřít dbm databázi

```
bool dbmclose (int dbm_identifier)
```

Odemkne a zavře určenou databázi.

dbmexists (PHP 3, PHP 4)

Zjistí, jestli pro zadaný klíč existuje v DBM databázi hodnota

```
bool dbmexists (int dbm_identifier, string key)
```

Vrací `true`, pokud existuje hodnota spojená s *`key`*.

dbmfetch (PHP 3, PHP 4)

Získat z DBM databáze hodnotu spojenou s určitým klíčem

```
string dbmfetch (int dbm_identifier, string key)
```

Vrací hodnotu spojenou s *`key`*.

dbminsert (PHP 3, PHP 4)

Vložit do DBM databáze hodnotu a klíč

```
int dbminsert (int dbm_identifier, string key, string value)
```

Přidá do databáze hodnotu s určeným klíčem.

Vrací -1, pokud byla databáze otevřena pouze pro čtení, 0, pokud bylo vložení úspěšné, a 1, pokud už určený klíč existuje. (K nahražení hodnoty použijte **dbmreplace()**.)

dbmreplace (PHP 3, PHP 4)

Nahredit v DBM databázi hodnotu s určitým klíčem

```
bool dbmreplace (int dbm_identifier, string key, string value)
```

Nahradí v databázi hodnotu spojenou s určeným klíčem.

Pokud tento klíče v databázi neexistuje, přidá ho.

dbmdelete (PHP 3, PHP 4)

Smazat v DMB databázi hodnotu spojenou s určitým klíčem

```
bool dbmdelete (int dbm_identifier, string key)
```

Vymaže z databáze hodnotu s klíčem *key*.

Pokud tento klíč v databázi neexistuje, vrací `false`.

dbmfirstkey (PHP 3, PHP 4)

Získat z DBM databáze první klíč

```
string dbmfirstkey (int dbm_identifier)
```

Vrací první klíč v databázi. Pozn.: Není zaručeno žádné pořadí, protože databáze může být vytvořena pomocí hash tabulky, což nezaručuje žádné řazení.

dbmnextkey (PHP 3, PHP 4)

Získat další klíč z DBM databáze

```
string dbmnextkey (int dbm_identifier, string key)
```

Vrací klíč následující po *key*. Zavoláním **dbmfirstkey()** následovaným postupným voláním **dbmnextkey()** se dají získat všechny key/value páry v DBM databázi. Například:

Příklad 1. Získání všech key/value párů v DBM databázi

```
$key = dbmfirstkey ($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch ($dbm_id, $key) . "\n";
    $key = dbmnextkey ($dbm_id, $key);
}
```

dblist (PHP 3, PHP 4)

Získat název používané DBM-kompatibilní knihovny

```
string dblist (void)
```

XVIII. dbx functions

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. In order to have these functions available, you must compile PHP with dbx support by using the `-enable-dbx` option and all options for the databases that will be used, e.g. for MySQL you must also specify `-with-mysql`. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases. To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, MySQL, PostgreSQL and ODBC are supported, but others will follow (soon, I hope :-).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

dbx_close (PHP 4 CVS only)

Close an open connection/database

```
boolean dbx_close (dbx_link_object link_identifier)
```

Returns TRUE on success, FALSE on error.

Příklad 1. dbx_close() example

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password")
    or die ("Could not connect");
print("Connected successfully");
dbx_close($link);
?>
```

Poznámka: Always refer to the module-specific documentation as well.

See also: **dbx_connect()**.

dbx_connect (PHP 4 CVS only)

Open a connection/database

```
dbx_link_object dbx_connect (string module, string host, string database, string
username, string password [, int persistent])
```

Returns: a dbx_link_object on success, FALSE on error. If a connection can be made but the database could not be selected, the function still returns a dbx_link_object. The 'persistent' parameter can be set to DBX_PERSISTENT so a persistent connection will be created.

Possible module names are given below, but keep in mind that they only work if the module is actually loaded.

- module 1: "mysql"
- module 2: "odbc"
- module 3: "pgsql"

The pgsql support is still experimental, and you should compile the actual pgsql module yourself after modifying one of the source files, otherwise you will get PostgreSQL warnings for every query.

The dbx_link_object has three members, a 'handle', a 'module' and a 'database'. The 'database' member is the name of the currently selected database. The 'module' member is for internal use by dbx only, and is actually the module number mentioned above. The 'handle' member is a valid handle for the connected database, and as such can be used in module-specific functions (if required), e.g.

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) would be better here
?>
```

Host, database, username and password parameters are expected, but not always used, depending on the connect-functions for the abstracted module.

Příklad 1. dbx_connect() example

```
<?php
$link = dbx_connect ("odbc", "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Could not connect");
print ("Connected successfully");
dbx_close ($link);
?>
```

Poznámka: Always refer to the module-specific documentation as well.

See also: **dbx_close()**.

dbx_error (PHP 4 CVS only)

Report the error message of the latest function call in the module (not just in the connection)

```
string dbx_error (dbx_link_object link_identifier)
```

Returns a string containing the error-message from the last function call of the module (e.g. mysql-module). If there are multiple connections on the same module, just the last error is given. If there are connections on different modules, the latest error is returned for the specified module (specified by the link parameter, that is). Note that the ODBC-module doesn't support an error_reporting function at the moment.

Příklad 1. dbx_error() example

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "select id from nonexistingtbl");
if ($result==0) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Poznámka: Always refer to the module-specific documentation as well.

dbx_query (PHP 4 CVS only)

Send a query and fetch all results (if any)

```
dbx_result_object dbx_query (dbx_link_object link_identifier, string sql_statement [,
long flags])
```

Returns a dbx_result_object or 1 on success (a result object is only returned for sql-statements that return results) or 0 on failure. The *flags* parameter is used to control the amount of information that is returned. It may be any combination of the constants DBX_RESULT_INFO, DBX_RESULT_INDEX, DBX_RESULT_ASSOC, OR-ed together. DBX_RESULT_INFO provides info about columns, such as field names and field types. DBX_RESULT_INDEX returns the results in a 2d indexed array (e.g. data[2][3], where 2 is the row (or record) number and 3 is the column (or field) number), where the first row and column are indexed at 0. DBX_RESULT_ASSOC associates the column indices with field names. Note that DBX_RESULT_INDEX is always

returned, regardless of the *flags* parameter. If DBX_RESULT_ASSOC is specified, DBX_RESULT_INFO is also returned even if it wasn't specified. This means that effectively only the combinations DBX_RESULT_INDEX, DBX_RESULT_INDEX | DBX_RESULT_INFO and DBX_RESULT_INDEX | DBX_RESULT_INFO | DBX_RESULT_ASSOC are possible. This last combination is the default if the *flags* parameter isn't specified. Associated results are actual references to the indexed data, so if you modify `data[0][0]`, then `data[0]['fieldnameforfirstcolumn']` is modified as well.

A dbx_result_object has five members (possibly four depending on *flags*), 'handle', 'cols', 'rows', 'info' (optional) and 'data'. Handle is a valid result identifier for the specified module, and as such can be used in module-specific functions, as seen in the example:

```
$result = dbx_query ($link, "SELECT id FROM tbl");
mysql_field_len ($result->handle, 0);
```

The cols and rows members contain the number of columns (or fields) and rows (or records) respectively, e.g.

```
$result = dbx_query ($link, "SELECT id FROM tbl");
echo "result size: " . $result->rows . " x " . $result->cols . "<br>\n";
```

The info member is only returned if DBX_RESULT_INFO and/or DBX_RESULT_ASSOC are specified in the *flags* parameter. It is a 2d array, that has two named rows ("name" and "type") to retrieve column information, e.g.

```
$result = dbx_query ($link, "SELECT id FROM tbl");
echo "column name: " . $result->info["name"][0] . "<br>\n";
echo "column type: " . $result->info["type"][0] . "<br>\n";
```

The data member contains the actual resulting data, possibly associated with column names as well. If DBX_RESULT_ASSOC is set, it is possible to use `$result->data[2]["fieldname"]`.

Příklad 1. dbx_query() example

```
<?php
$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl");
if ($result==0) echo "Query failed\n<br>";
elseif ($result==1) {
    echo "Query executed successfully\n<br>";
} else {
    $rows=$result->rows;
    $cols=$result->cols;
    echo "<p>table dimension: {$result->rows} x {$result->cols}<br><table border=1>\n";
    echo "<tr>";
    for ($col=0; $col<$cols; ++$col) {
        echo "<td>-{$result->info["name"][$col]}-<br>-{$result->info["type"][$col]}-
</td>";
    }
    echo "</tr>\n";
    for ($row=0; $row<$rows; ++$row){
        echo "<tr>";
        for ($col=0; $col<$cols; ++$col) {
            echo "<td>-{$result->data[$row][$col]}-</td>";
        }
        echo "</tr>\n";
    }
    echo "</table><p>\n";
    echo "table dimension: {$result-
>rows} x id, parentid, description<br><table border=1>\n";
```

```
    for ($row=0; $row<$rows; ++$row) {
        echo "<tr>";
        echo "<td>-{$result->data[$row]["id"]}-</td>";
        echo "<td>-{$result->data[$row]["parentid"]}-</td>";
        echo "<td>-{$result->data[$row]["description"]}-</td>";
        echo "</tr>\n";
    }
    echo "</table><p>\n";
}
dbx_close($link);
?>
```

Poznámka: Always refer to the module-specific documentation as well.

See also: **dbx_connect()**.

dbx_sort (PHP 4 CVS only)

Sort a result from a dbx_query by a custom sort function

```
boolean dbx_sort (dbx_result_object result, string user_compare_function)
```

Returns TRUE on success, FALSE on error.

Příklad 1. dbx_sort() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_cmp_asc ($a, $b, "parentid");
    if (!$rv) $rv = dbx_cmp_asc ($a, $b, "id");
    return $rv;
}

$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "resulting data is now ordered by id<br>";
dbx_query ($result, "user_re_order");
echo "resulting data is now ordered by parentid, then by id<br>";
dbx_close ($link);
?>
```

See also **dbx_cmp_asc()** and **dbx_cmp_desc()**.

dbx_cmp_asc (PHP 4 CVS only)

Compare two rows for sorting in ascending order

```
int dbx_cmp_asc (array row_a, array row_b, string columnname_or_index)
```

Returns 0 if row_a[$columnname_or_index] is equal to row_b[$columnname_or_index], 1 if it is greater and -1 if it is smaller.

Příklad 1. dbx_cmp_asc() example

```
<?php
```

```
function user_re_order ($a, $b) {
    $rv = dbx_cmp_asc ($a, $b, "parentid");
    if (!$rv) {
        $rv = dbx_cmp_asc ($a, $b, "id");
        return $rv;
    }
}

$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "resulting data is now ordered by id<br>";
dbx_query ($result, "user_re_order");
echo "resulting data is now ordered by parentid, then by id<br>";
dbx_close ($link);
?>
```

See also **dbx_sort()** and **dbx_cmp_desc()**.

dbx_cmp_desc (PHP 4 CVS only)

Compare two rows for sorting in descending order

```
int dbx_cmp_desc (array row_a, array row_b, string columnname_or_index)
```

Returns 0 if row_a[$columnname_or_index] is equal to row_b[$columnname_or_index], -1 if it is greater and 1 if it is smaller.

Příklad 1. dbx_cmp_desc() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_cmp_asc ($a, $b, "parentid");
    if (!$rv) {
        $rv = dbx_cmp_asc($a, $b, "id");
        return $rv;
    }
}

$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "resulting data is now ordered by id<br>";
dbx_query ($result, "user_re_order");
echo "resulting data is now ordered by parentid, then by id<br>";
dbx_close ($link);
?>
```

See also **dbx_sort()** and **dbx_cmp_asc()**.

XIX. Adresářové funkce

chdir (PHP 3, PHP 4)

change directory

```
int chdir (string directory)
```

Mění aktuální adresář PHP na *directory*. Pokud se nepodařilo změnit adresář, vrací FALSE, jinak TRUE.

dir (PHP 3, PHP 4)

directory class

```
new dir (string directory)
```

Pseudo-objektově orientovaný mechanismus pro čtení adresáře. Otevře zadaný *directory*. Po otevření adresáře jsou přístupné dvě vlastnosti. Vlastnost handle je použitelná s jinými adresářovými funkcemi jako **readdir()**, **rewinddir()** a **closedir()**. Hodnotou vlastnosti path je cesta k otevřenému adresáři. Dále jsou přístupné tři metody: read, rewind a close.

Příklad 1. Dir() příklad

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
    echo $entry."<br>\n";
}
$d->close();
```

closedir (PHP 3, PHP 4)

Zavírá otevřený adresář

```
void closedir (int dir_handle)
```

Zavírá proud z adresáře specifikovaný v *dir_handle*. Proud musí pocházet z funkce **opendir()**.

getcwd (PHP 4 >= 4.0b4)

Získává aktuální pracovní adresář

```
string getcwd(void);
```

Vrací aktuální pracovní adresář.

opendir (PHP 3, PHP 4)

Otevře adresář

```
int opendir (string path)
```

Vrací deskriptor adresáře použitelný v následných voláních **closedir()**, **readdir()**, a **rewinddir()**.

readdir (PHP 3, PHP 4)

Přečte položku z deskriptoru adresáře

```
string readdir (int dir_handle)
```

Vrací název dalšího souboru v adresáři. Názvy souborů nejsou nijak tříděny.

Příklad 1. Vypiš všechny soubory v adresáři

```
// Note that !== did not exist until 4.0.0-RC2
<?php
$handle=opendir('.');
echo "Directory handle: $handle\n";
echo "Files:\n";
while (($file = readdir($handle))!==false) {
    echo "$file\n";
}
closedir($handle);
?>
```

Všimněte si, že **readdir()** vráží také . a .. položky. Pokud je nechcete, můžete je snadno vynechat:

Příklad 2. Vypiš všechny soubory v adresáři a vynech . a ..

```
<?php
$handle=opendir('.');
while (false!==($file = readdir($handle))) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
    }
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4)

rewind directory handle

```
void rewinddir (int dir_handle)
```

Resets the directory stream indicated by *dir_handle* to the beginning of the directory.

XX. DOM XML funkce

Tyto funkce jsou přístupné pouze pokud bylo PHP zkonfigurováno s volbou `-with-dom=[DIR]`, s využitím GNOME xml knihovny. Je potřeba nejméně libxml-2.0.0 (ne betaverze). Tyto funkce byly přidány v PHP 4.

Tento modul definuje následující konstanty:

Tabulka 1. XML konstanty

Konstanta	Hodnota	Popis
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Tento modul definuje několik tříd. DOM XML funkce vrací rozparsovaný strom XML dokumentu, kde každý uzel je objektem patrícím do jedné z těchto tříd.

xmldoc (PHP 4 >= 4.0b4)

Vytvořit DOM objekt z XML dokumentu

```
object xmldoc (string str)
```

Tato funkce parsuje XML dokument v *str* a vrací objekt třídy "Dom document", který má valstnosti "doc" (resource), "version" (string) and "type" (long).

xmldocfile (PHP 4 >= 4.0b4)

Vytvořit DOM objekt z XML souboru

```
object xmldocfile (string filename)
```

Parsuje XML dokument v souboru pojmenovaném *filename* a vrací objekt třídy "Dom document", který má vlastnosti "doc" (resource), "version" (string).

xmltree (PHP 4 >= 4.0b4)

Vytvořit strom PHP objektů z XML dokumentu

```
object xmltree (string str)
```

Parsuje XML dokument v *str* a vrací strom PHP objektů.

XXI. Error Handling and Logging Functions

These are functions dealing with error handling and logging. They allow you to define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs.

With the logging functions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

The error reporting functions allow you to customize what level and kind of error feedback is given, ranging from simple notices to customized functions returned during errors.

error_log (PHP 3, PHP 4)

send an error message somewhere

```
int error_log (string message, int message_type [, string destination [, string
extra_headers]])
```

Sends an error message to the web server's error log, a TCP port or to a file. The first parameter, *message*, is the error message that should be logged. The second parameter, *message_type* says where the message should go:

Tabulka 1. error_log() log types

0	*message is sent to PHP's system logger, using the Operating System's system logging mechanism or a file, depending on what the error_log configuration directive is set to.*
1	*message is sent by email to the address in the destination parameter. This is the only message type where the fourth parameter, extra_headers is used. This message type uses the same internal function as* ***Mail() does.***
2	*message is sent through the PHP debugging connection. This option is only available if remote debugging has been enabled. In this case, the destination parameter specifies the host name or IP address and optionally, port number, of the socket receiving the debug information.*
3	*message is appended to the file destination.*

Příklad 1. error_log() examples

```
// Send notification through the server log if we can not
// connect to the database.
if (!Ora_Logon ($username, $password)) {
    error_log ("Oracle database not available!", 0);
}

// Notify administrator by email if we run out of FOO
if (!($foo = allocate_new_foo()) {
    error_log ("Big trouble, we're all out of FOOs!", 1,
               "operator@mydomain.com");
}

// other ways of calling error_log():
error_log ("You messed up!", 2, "127.0.0.1:7000");
error_log ("You messed up!", 2, "loghost");
error_log ("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4)

set which PHP errors are reported

```
int error_reporting ([int level])
```

Sets PHP's error reporting level and returns the old level. The error reporting level is either a bitmask, or named constant. Using named constants is strongly encouraged to ensure compatibility for future versions. As error levels are added, the range of integers increases, so older integer-based error levels will not always behave as expected.

Příklad 1. Error Integer changes

```
error_reporting (55);   // PHP 3 equivalent to E_ALL ^ E_NOTICE

/* ...in PHP 4, '55' would mean (E_ERROR | E_WARNING | E_PARSE |
E_CORE_ERROR | E_CORE_WARNING) */

error_reporting (2039); // PHP 4 equivalent to E_ALL ^ E_NOTICE

error_reporting (E_ALL ^ E_NOTICE); // The same in both PHP 3 and 4
```

Follow the links for the internal values to get their meanings:

Tabulka 1. error_reporting() bit values

constant	value
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE

Příklad 2. error_reporting() examples

```
error_reporting(0);
/* Turn off all reporting */

error_reporting (7); // Old syntax, PHP 2/3
error_reporting  (E_ERROR | E_WARNING | E_PARSE); // New syntax for PHP 3/4
/* Good to use for simple running errors  */

error_reporting  (15); // Old syntax, PHP 2/3
error_reporting (E_ERROR | E_WARNING | E_PARSE | E_NOTICE); // New syntax for PHP 3/4
/*  good for code authoring to report uninitialized or (possibly mis-
spelled) variables */

error_reporting (63); // Old syntax, PHP 2/3
error_reporting (E_ALL); // New syntax for PHP 3/4
/* report all PHP errors */
```

restore_error_handler (PHP 4 >= 4.0.1)

Restores the previous error handler function

```
void restore_error_handler (void)
```

Used after changing the error handler function using **set_error_handler()**, to revert to the previous error handler (which could be the built-in or a user defined function)

See also **error_reporting()**, **set_error_handler()**, **trigger_error()**, **user_error()**

set_error_handler (PHP 4 >= 4.0.1)

Sets a user-defined error handler function.

```
string set_error_handler (string error_handler)
```

Sets a user function (*error_handler*) to handle errors in a script. Returns the previously defined error handler (if any), or false on error. This function can be used for defining your own way of handling errors during runtime, for example in applications in which you need to do cleanup of data/files when a critical error happens, or when you need to trigger an error under certain conditions (using **trigger_error()**)

The user function needs to accept 2 parameters: the error code, and a string describing the error. From PHP 4.0.2, an additional 3 optional parameters are supplied: the filename in which the error occured, the line number in which the error occured, and the context in which the error occured (an array that points to the active symbol table at the point the error occurred).

The example below shows the handling of internal execptions by triggering errors and handling them with a user defined function:

Příklad 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP 4 only
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);

// set the error reporting level for this script
error_reporting (FATAL | ERROR | WARNING);

// error handler function
function myErrorHandler ($errno, $errstr, $errfile, $errline) {
  switch ($errno) {
  case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".$errline." of file ".$errfile;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
  case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
  case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
```

```
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
  }
}

// function to test the error handling
function scale_by_log ($vect, $scale) {
  if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
  if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
  }
  for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
      trigger_error("Value at position $i is not a number, using 0 (zero)",
        WARNING);
    $temp[$i] = log($scale) * $vect[$i];
  }
  return $temp;
}

// set to the user defined error handler
$old_error_handler = set_error_handler("myErrorHandler");

// trigger some errors, first define a mixed array with a non-numeric item
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// now generate second array, generating a warning
echo "---\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// this is trouble, we pass a string instead of an array
echo "---\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// this is a critical error, log of zero or negative number is undefined
echo "---\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

And when you run this sample script, the output will be

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
---
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
```

```
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
---
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
---
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br>
  Fatal error in line 36 of file trigger_error.php, PHP 4.0.2 (Linux)<br>
Aborting...<br>
```

It is important to remember that the standard PHP error handler is completely bypassed. **error_reporting()** settings will have no effect and your error handler will be called regardless - however you are still able to read the current value of **error_reporting()** and act appropriately. Of particular note is that this value will be 0 if the statement that caused the error was prepended by the @ error-control operator.

Also note that it is your responsibility to **die()** if necessary. If the error-handler function returns, script execution will continue with the next statement after the one that caused an error.

See also **error_reporting()**, **restore_error_handler()**, **trigger_error()**, **user_error()**

trigger_error (PHP 4 >= 4.0.1)

Generates a user-level error/warning/notice message

```
void trigger_error (string error_msg [, int error_type])
```

Used to trigger a user error condition, it can be used by in conjunction with the built-in error handler, or with a user defined function that has been set as the new error handler (**set_error_handler()**). It only works with the E_USER family of constants, and will default to `E_USER_NOTICE`.

This function is useful when you need to generate a particular response to an exception at runtime. For example:

```
if (assert ($divisor == 0))
   trigger_error ("Cannot divide by zero", E_USER_ERROR);
```

Poznámka: See **set_error_handler()** for a more extensive example.

See also **error_reporting()**, **set_error_handler()**, **restore_error_handler()**, **user_error()**

user_error (PHP 4 >= 4.0RC2)

Generates a user-level error/warning/notice message

```
void user_error (string error_msg [, int error_type])
```

This is an alias for the function **trigger_error()**.

See also **error_reporting()**, **set_error_handler()**, **restore_error_handler()**, and **trigger_error()**

XXII. filePro funkce

Tyto funkce umožňují read-only (pouze pro čtení) přístup k datům uloženým ve filePro databázích.

filePro je registrovaná obchodní značka fP Technologies, Inc. Více informací o filePro najdete na http://www.fptech.com/.

filepro (PHP 3, PHP 4)

Přečíst a ověřit mapový soubor

```
bool filepro (string directory)
```

Přečte a ověří mapový soubor a uloží počet sloupců a info.
Nedochází k zamykání, takže byste se měli vyhnout úpravám vaší filePro databáze, pokud může být otevřena v PHP.

filepro_fieldname (PHP 3, PHP 4)

Zjistit název sloupce

```
string filepro_fieldname (int field_number)
```

Vrátí název sloupce odpovídajícího *`field_number`*.

filepro_fieldtype (PHP 3, PHP 4)

Zjistit typ sloupce

```
string filepro_fieldtype (int field_number)
```

Vrátí editační typ sloupce odpovídajícího *`field_number`*.

filepro_fieldwidth (PHP 3, PHP 4)

Zjistit šířku sloupce

```
int filepro_fieldwidth (int field_number)
```

Vrátí šířku sloupce odpovídajícího *`field_number`*.

filepro_retrieve (PHP 3, PHP 4)

Získat data z filePro databáze

```
string filepro_retrieve (int row_number, int field_number)
```

Vrátí data z určeného místa v databázi.

filepro_fieldcount (PHP 3, PHP 4)

Zjistit, kolik sloupců je ve filePro databázi

```
int filepro_fieldcount(void);
```

Vrátí počet polí (sloupců) v otevřené filePro databázi.

Viz také **filepro()**.

filepro_rowcount (PHP 3, PHP 4)

Zjistit, kolik řádků je ve filePro databázi

```
int filepro_rowcount(void);
```

Vrátí počet řádků v otevřené filePro databázi.

Viz také **filepro()**.

XXIII. Filesystémové funkce

basename (PHP 3, PHP 4)

Vrátit část cesty obsahující název souboru

```
string basename (string path)
```

Přijímá řetězec obsahující cestu na soubor, vrací název souboru.

Na Windows, se jako oddělovač cesty dá použít jak lomítko (/), tak zpětné lomítko (\). V ostatních prostředích je to normální lomítko (/).

Příklad 1. Ukázka basename()

```
$path = '/home/httpd/html/index.php3' ;
$file = basename( $path ) ; // $file má hodnotu "index.php3"
```

Viz také **dirname()**

chgrp (PHP 3, PHP 4)

Změnit skupinu souboru

```
int chgrp (string filename, mixed group)
```

Pokusí se nastavit skupinu souboru *filename* na *group*. Pouze superuser může libovolně změnit skupinu souboru; ostatní uživatelé mohou změnit skupinu souboru na jinou skupinu, jíž jsou členy.

Při úspěchu vrací true; jinak false.

Viz také **chown()** a **chmod()**.

Poznámka: Tato funkce nefunguje na Windows systémech.

chmod (PHP 3, PHP 4)

Změnit mód souboru

```
int chmod (string filename, int mode)
```

Pokusí se změnit mód souboru *filename* na *mode*.

Pozn.: *mode* se nepovažuje automaticky za oktalovou hodnotu, takže řetězce (jako "g+w") nebudou správně fungovat. Pokud si chcete zajistit očekávané chování, musíte na začátek *mode* přidat nulu (0):

```
chmod ("/somedir/somefile", 755);   // desitkove cislo; zrejme nespravne
chmod ("/somedir/somefile", "u+rwx,go+rx"); // retezec; nespravny
chmod ("/somedir/somefile", 0755);  // oktal; spravna hodnota modu
```

Při úspěchu vrací true, jinak false.

Viz také **chown()** a **chgrp()**.

Poznámka: Tato funkce nefunguje na Windows systémech

chown (PHP 3, PHP 4)

Změnit majitele souboru

```
int chown (string filename, mixed user)
```

Pokusí se změnit majitele souboru na *user*. Pouze superuser může změnit majitele souboru.

Při úspěchu vrací `true`, jinak `false`.

Viz také **chown()** a **chmod()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

clearstatcache (PHP 3, PHP 4)

Vymaže cache stavu souborů

```
void clearstatcache(void);
```

Volání systémových funkcí stat či lstat má docela velkou režii. Tudíž se výsledek posledního volání všech status funkcí (viz seznam níže) ukládá pro použití při dalším takovém volání používajícím stejný název souboru. Pokud chcete vynutit novou kontrolu stavu, např. pokud je soubor kontrolován mnohokrát a může se změnit nebo zmizet, použijte tuto funkci k uvolnění výsledků posledního volání z paměti.

Tato hodnota se cachuje pouze po dobu běhu skriptu.

Ovlivněné funkce: **stat()**, **lstat()**, **file_exists()**, **is_writable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, a **fileperms()**.

copy (PHP 3, PHP 4)

Zkopírovat soubor

```
int copy (string source, string dest)
```

Vytvoří kopii souboru. Vrací `true` pokud se kopírování zdařilo, jinak `false`.

Příklad 1. Ukázka copy()

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
```

Viz také **rename()**.

delete (unknown)

Falešná položka manuálu

```
void delete (string file)
```

Toto je falešná položka manuálu, která by měla pomoci lidem, kteří hledají **unlink()** nebo **unset()** na špatném místě.

Viz také **unlink()** (mazání souborů), **unset()** (mazání proměnných).

dirname (PHP 3, PHP 4)

Vrací část cesty obsahující název adresáře

```
string dirname (string path)
```

Přijímá řetězec obsahující cestu na soubor a vrací název adresáře.

Na Windows se jako oddělovač sesty dají používat jak lomítko (/), tak zpětné lomítko (\). Na ostatních systémech je to lomítko (/).

Příklad 1. Ukázka dirname()

```
$path = "/etc/passwd";
$file = dirname ($path); // $file obsahuje "/etc"
```

Viz také **basename()**

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Vrátit diskový prostor dostupný v adresáři

```
float diskfreespace (string directory)
```

Přijímá řetězec obsahující cestu na adresář, vrací počet bytů dostupných na odpovídajícím filesystému nebo oddílu.

Příklad 1. Ukázka diskfreespace()

```
$df = diskfreespace("/"); // $df obsahuje pocet bytu
                          // dostupnych na "/"
```

fclose (PHP 3, PHP 4)

Zavřít otevřený deskriptor souboru

```
int fclose (int fp)
```

Soubor, na který *fp* ukazuje, se zavře.

Vrací `true` při úspěchu a `false` při selhání.

Deskriptor souboru musí být platný, a musí ukazovat na soubor úspěšně otevřený pomocí **fopen()** nebo **fsockopen()**.

feof (PHP 3, PHP 4)

Tests for end-of-file on a file pointer

```
int feof (int fp)
```

Returns `true` if the file pointer is at EOF or an error occurs; otherwise returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

fflush (PHP 4 >= 4.0.1)

Flushes the output to a file

```
int fflush (int fp)
```

This function forces a write of all buffered output to the to the resource pointed to by the file handle *fp*. Returns `true` if succesful, false otherwise.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

fgetc (PHP 3, PHP 4)

Gets character from file pointer

```
string fgetc (int fp)
```

Returns a string containing a single character read from the file pointed to by fp. Returns FALSE on EOF.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

Viz také **fread()**, **fopen()**, **popen()**, **fsockopen()**, and **fgets()**.

fgetcsv (PHP 3>= 3.0.8, PHP 4)

Gets line from file pointer and parse for CSV fields

```
array fgetcsv (int fp, int length [, string delimiter])
```

Similar to **fgets()** except that **fgetcsv()** parses the line it reads for fields in CSV format and returns an array containing the fields read. The field delimiter is a comma, unless you specify another delimiter with the optional third parameter.

Fp must be a valid file pointer to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**

Length must be greater than the longest line to be found in the CSV file (allowing for trailing line-end characters).

Fgetcsv() returns `false` on error, including end of file.

N.B. A blank line in a CSV file will be returned as an array comprising a single null field, and will not be treated as an error.

Příklad 1. Fgetcsv() example - Read and print entire contents of a CSV file

```
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num fields in line $row: <br>";
```

```
    $row++;
    for ($c=0; $c<$num; $c++) {
        print $data[$c] . "<br>";
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4)

Gets line from file pointer

```
string fgets (int fp, int length)
```

Returns a string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline (which is included in the return value), or on EOF (whichever comes first).

If an error occurs, returns false.

Common Pitfalls:

People used to the 'C' semantics of fgets should note the difference in how EOF is returned.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

A simple example follows:

Příklad 1. Reading a file line by line

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof ($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Viz také **fread()**, **fopen()**, **popen()**, **fgetc()**, **fsockopen()**, and **socket_set_timeout()**.

fgetss (PHP 3, PHP 4)

Gets line from file pointer and strip HTML tags

```
string fgetss (int fp, int length [, string allowable_tags])
```

Identical to **fgets()**, except that fgetss attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Poznámka: *allowable_tags* was added in PHP 3.0.13, PHP4B3.

Viz také **fgets()**, **fopen()**, **fsockopen()**, **popen()**, and **strip_tags()**.

file (PHP 3, PHP 4)

Reads entire file into an array

```
array file (string filename [, int use_include_path])
```

Identical to **readfile()**, except that **file()** returns the file in an array. Each element of the array corresponds to a line in the file, with the newline still attached.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

```
<?php
// get a web page into an array and print it out
$fcontents = file ('http://www.php.net');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " . htmlspecialchars ($line) . "<br>\n";
}

// get a web page into a string
$fcontents = join ('', file ('http://www.php.net'));
?>
```

Viz také **readfile()**, **fopen()**, **fsockopen()**, and **popen()**.

file_exists (PHP 3, PHP 4)

Checks whether a file exists

```
bool file_exists (string filename)
```

Returns `true` if the file specified by *`filename`* exists; `false` otherwise.

file_exists() will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See **clearstatcache()** for more details.

fileatime (PHP 3, PHP 4)

Gets last access time of file

```
int fileatime (string filename)
```

Returns the time the file was last accessed, or `false` in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See **clearstatcache()** for more details.

Note: The atime of a file is supposed to change whenever the data blocks of a file are being read. This can be costly performancewise when an appliation regularly accesses a very large number of files or directories. Some Unix filesystems can be mounted with atime updates disabled to increase the performance of such applications; USENET news spools are a common example. On such filesystems this function will be useless.

filectime (PHP 3, PHP 4)

Gets inode change time of file

```
int filectime (string filename)
```

Returns the time the file was last changed, or `false` in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See **clearstatcache()** for more details.

Note: In most Unix filesystem, a file is considered changed, when it’s Inode data is changed, that is, when the permissions, the owner, the group or other metadata from the Inode is written to. Viz také **filemtime()** (this is what you want to use when you want to create "Last Modified" footers on web pages) and **fileatime()**.

Note: In some Unix texts the ctime of a file is being referred to as the creation time of the file. This is wrong. There is no creation time for Unix files in most Unix filesystems.

filegroup (PHP 3, PHP 4)

Gets file group

```
int filegroup (string filename)
```

Returns the group ID of the owner of the file, or `false` in case of an error. The group ID is returned in numerical format, use **posix_getgrgid()** to resolve it to a group name.

The results of this function are cached. See **clearstatcache()** for more details.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

fileinode (PHP 3, PHP 4)

Gets file inode

```
int fileinode (string filename)
```

Returns the inode number of the file, or `false` in case of an error.

The results of this function are cached. See **clearstatcache()** for more details.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

filemtime (PHP 3, PHP 4)

Gets file modification time

```
int filemtime (string filename)
```

Returns the time the file was last modified, or `false` in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See **clearstatcache()** for more details.

Note: This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed. Use **date()** on the result of this function to get a printable modification date for use in page footers.

fileowner (PHP 3, PHP 4)

Gets file owner

```
int fileowner (string filename)
```

Returns the user ID of the owner of the file, or `false` in case of an error. The user ID is returned in numerical format, use **posix_getpwuid()** to resolve it to a username.

The results of this function are cached. See **clearstatcache()** for more details.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

fileperms (PHP 3, PHP 4)

Gets file permissions

```
int fileperms (string filename)
```

Returns the permissions on the file, or `false` in case of an error.

The results of this function are cached. See **clearstatcache()** for more details.

filesize (PHP 3, PHP 4)

Gets file size

```
int filesize (string filename)
```

Returns the size of the file, or `false` in case of an error.

The results of this function are cached. See **clearstatcache()** for more details.

filetype (PHP 3, PHP 4)

Gets file type

```
string filetype (string filename)
```

Returns the type of the file. Possible values are fifo, char, dir, block, link, file, and unknown.

Returns false if an error occurs.

The results of this function are cached. See **clearstatcache()** for more details.

flock (PHP 3>= 3.0.7, PHP 4)

Portable advisory file locking

```
bool flock (int fp, int operation [, int wouldblock])
```

PHP supports a portable way of locking complete files in an advisory way (which means all accessing programs have to use the same way of locking or it will not work).

flock() operates on *fp* which must be an open file pointer. *operation* is one of the following values:

- To acquire a shared lock (reader), set *operation* to LOCK_SH (set to 1 prior to PHP 4.0.1).

- To acquire an exclusive lock (writer), set *operation* to LOCK_EX (set to 2 prior to PHP 4.0.1).
- To release a lock (shared or exclusive), set *operation* to LOCK_UN (set to 3 prior to PHP 4.0.1).
- If you don't want **flock()** to block while locking, add LOCK_NB (4 prior to PHP 4.0.1) to *operation*.

Flock() allows you to perform a simple reader/writer model which can be used on virtually every platform (including most Unices and even Windows). The optional 3rd argument is set to `true` if the lock would block (EWOULDBLOCK errno condition)

Flock() returns `true` on success and `false` on error (e.g. when a lock could not be acquired).

> **Varování**
>
> On most operation systems **flock()** is implemented at the process level. When using a multithreaded server API like ISAPI you cannot rely on **flock()** to protect files against other PHP scripts running in parallel threads of the same server instance!

fopen (PHP 3, PHP 4)

Opens file or URL

```
int fopen (string filename, string mode [, int use_include_path])
```

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and a file pointer is returned to the beginning of the text of the response. A 'Host:' header is sent with the request in order to handle name-based virtual hosts.

Does not handle HTTP redirects, so you must include trailing slashes on directories.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and a pointer to the requested file is returned. If the server does not support passive mode ftp, this will fail. You can open files for either reading or writing via ftp (but not both simultaneously).

If *filename* is one of "php://stdin", "php://stdout", or "php://stderr", the corresponding stdio stream will be opened. (This was introduced in PHP 3.0.13; in earlier versions, a filename such as "/dev/stdin" or "/dev/fd/0" must be used to access the stdio streams.)

If *filename* begins with anything else, the file will be opened from the filesystem, and a file pointer to the file opened is returned.

If the open fails, the function returns false.

mode may be any of the following:

- 'r' - Open for reading only; place the file pointer at the beginning of the file.
- 'r+' - Open for reading and writing; place the file pointer at the beginning of the file.
- 'w' - Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'w+' - Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'a' - Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+' - Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

The *mode* may contain the letter 'b'. This is useful only on systems which differentiate between binary and text files (i.e., it's useless on Unix). If not needed, this will be ignored.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Příklad 1. Fopen() example

```
$fp = fopen ("/home/rasmus/file.txt", "r");
$fp = fopen ("/home/rasmus/file.gif", "wb");
$fp = fopen ("http://www.php.net/", "r");
$fp = fopen ("ftp://user:password@example.com/", "w");
```

If you are experiencing problems with reading and writing to files and you're using the server module version of PHP, remember to make sure that the files and directories you're using are accessible to the server process.

On the Windows platform, be careful to escape any backslashes used in the path to the file, or use forward slashes.

```
$fp = fopen ("c:\\data\\info.txt", "r");
```

Viz také **fclose()**, **fsockopen()**, **socket_set_timeout()**, and **popen()**.

fpassthru (PHP 3, PHP 4)

Output all remaining data on a file pointer

```
int fpassthru (int fp)
```

Reads to EOF on the given file pointer and writes the results to standard output.

If an error occurs, **fpassthru()** returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**. The file is closed when **fpassthru()** is done reading it (leaving *fp* useless).

If you just want to dump the contents of a file to stdout you may want to use the **readfile()**, which saves you the **fopen()** call.

Viz také **readfile()**, **fopen()**, **popen()**, and **fsockopen()**

fputs (PHP 3, PHP 4)

Writes to a file pointer

```
int fputs (int fp, string str [, int length])
```

Fputs() is an alias to **fwrite()**, and is identical in every way. Note that the *length* parameter is optional and if not specified the entire string will be written.

fread (PHP 3, PHP 4)

Binary-safe file read

```
string fread (int fp, int length)
```

Fread() reads up to *length* bytes from the file pointer referenced by *fp*. Reading stops when *length* bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a file into a string
```

```
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Viz také **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **fscanf()**, **file()**, and **fpassthru()**.

fscanf (PHP 4 >= 4.0.1)

Parses input from a file according to a format

```
mixed fscanf (int handle, string format [, string var1...])
```

The function **fscanf()** is similar to **sscanf()**, but it takes its input from a file associated with *handle* and interprets the input according to the specified *format*. If only two parameters were passed to this function, the values parsed will be returned as an array. Otherwise, if optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Příklad 1. Fscanf() Example

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... do something with the values
}
fclose($fp);
```

Příklad 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

Viz také **fread()**, **fgets()**, **fgetss()**, **sscanf()**, **printf()**, and **sprintf()**.

fseek (PHP 3, PHP 4)

Seeks on a file pointer

```
int fseek (int fp, int offset [, int whence])
```

Sets the file position indicator for the file referenced by *fp*.The new position, measured in bytes from the beginning of the file, is obtained by adding *offset* to the position specified by *whence*, whose values are defined as follows:

SEEK_SET - Set position equal to *offset bytes.*
SEEK_CUR - Set position to current location plus *offset.*
SEEK_END - Set position to end-of-file plus *offset.*

If *whence is not specified, it is assumed to be SEEK_SET.*

Upon success, returns 0; otherwise, returns -1. Pozn.: seeking past EOF is not considered an error.

May not be used on file pointers returned by **fopen()** if they use the "http://" or "ftp://" formats.

> **Poznámka:** The *whence* argument was added after PHP 4.0 RC1.

Viz také **ftell()** and **rewind()**.

fstat (PHP 4 >= 4.0RC1)

Gets information about a file using an open file pointer

```
array fstat (int fp)
```

Gathers the statistics of the file opened by the file pointer fp. This function is similar to the **stat()** function except that it operates on an open file pointer instead of a filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. number of links
4. user id of owner
5. group id owner
6. device type if inode device *
7. size in bytes
8. time of last access
9. time of last modification
10. time of last change
11. blocksize for filesystem I/O *
12. number of blocks allocated

* - only valid on systems supporting the st_blksize type–other systems (i.e. Windows) return -1

The results of this function are cached. See **clearstatcache()** for more details.

ftell (PHP 3, PHP 4)

Tells file pointer read/write position

```
int ftell (int fp)
```

Returns the position of the file pointer referenced by fp; i.e., its offset into the file stream.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()** or **popen()**.

Viz také **fopen()**, **popen()**, **fseek()** and **rewind()**.

ftruncate (PHP 4 >= 4.0RC1)

Truncates a file to a given length.

```
int ftruncate (int fp, int size)
```

Takes the filepointer, fp, and truncates the file to length, size. This function returns `true` on success and `false` on failure.

fwrite (PHP 3, PHP 4)

Binary-safe file write

```
int fwrite (int fp, string string [, int length])
```

fwrite() writes the contents of *`string`* to the file stream pointed to by *`fp`*. If the *`length`* argument is given, writing will stop after *`length`* bytes have been written or the end of *`string`* is reached, whichever comes first.

Pozn.: if the *`length`* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *`string`*.

Viz také **fread()**, **fopen()**, **fsockopen()**, **popen()**, and **fputs()**.

set_file_buffer (PHP 3>= 3.0.8, PHP 4 >= 4.0.1)

Sets file buffering on the given file pointer

```
int set_file_buffer (int fp, int buffer)
```

Output using **fwrite()** is normally buffered at 8K. This means that if there are two processess wanting to write to the same output stream (a file), each is paused after 8K of data to allow the other to write. **set_file_buffer()** sets the buffering for write operations on the given filepointer *`fp`* to *`buffer`* bytes. If *`buffer`* is 0 then write operations are unbuffered. This ensures that all writes with **fwrite()** are completed before other processes are allowed to write to that output stream.

The function returns 0 on success, or EOF if the request cannot be honored.

The following example demonstrates how to use **set_file_buffer()** to create an unbuffered stream.

Příklad 1. set_file_buffer() example

```
$fp=fopen($file, "w");
if($fp){
  set_file_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
}
```

Viz také **fopen()**, **fwrite()**.

is_dir (PHP 3, PHP 4)

Tells whether the filename is a directory

```
bool is_dir (string filename)
```

Returns `true` if the filename exists and is a directory.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_file()** and **is_link()**.

is_executable (PHP 3, PHP 4)

Tells whether the filename is executable

```
bool is_executable (string filename)
```

Returns `true` if the filename exists and is executable.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_file()** and **is_link()**.

is_file (PHP 3, PHP 4)

Tells whether the filename is a regular file

```
bool is_file (string filename)
```

Returns `true` if the filename exists and is a regular file.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_dir()** and **is_link()**.

is_link (PHP 3, PHP 4)

Tells whether the filename is a symbolic link

```
bool is_link (string filename)
```

Returns `true` if the filename exists and is a symbolic link.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_dir()** and **is_file()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

is_readable (PHP 3, PHP 4)

Tells whether the filename is readable

```
bool is_readable (string filename)
```

Returns `true` if the filename exists and is readable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_writable()**.

is_writable (PHP 4 >= 4.0b2)

Tells whether the filename is writable

```
bool is_writable (string filename)
```

Returns `true` if the filename exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See **clearstatcache()** for more details.

Viz také **is_readable()**.

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Tells whether the file was uploaded via HTTP POST.

```
bool is_uploaded_file (string filename)
```

This function is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2.

Returns `true` if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working–for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

Viz také **move_uploaded_file()**, and the section Handling file uploads for a simple usage example.

link (PHP 3, PHP 4)

Create a hard link

```
int link (string target, string link)
```

Link() creates a hard link.

Viz také the **symlink()** to create soft links, and **readlink()** along with **linkinfo()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

linkinfo (PHP 3, PHP 4)

Gets information about a link

```
int linkinfo (string path)
```

Linkinfo() returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by *path*) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns 0 or FALSE in case of error.

Viz také **symlink()**, **link()**, and **readlink()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech

mkdir (PHP 3, PHP 4)

Makes directory

```
int mkdir (string pathname, int mode)
```

Attempts to create the directory specified by pathname.

Pozn.: you probably want to specify the mode as an octal number, which means it should have a leading zero.

```
mkdir ("/path/to/my/dir", 0700);
```

Returns `true` on success and `false` on failure.

Viz také **rmdir()**.

move_uploaded_file (PHP 4 >= 4.0.3)

Moves an uploaded file to a new location.

```
bool move_uploaded_file (string filename, string destination)
```

This function is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2.

This function checks to ensure that the file designated by *filename* is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by *destination*.

If *filename* is not a valid upload file, then no action will occur, and **move_uploaded_file()** will return `false`.

If *filename* is a valid upload file, but cannot be moved for some reason, no action will occur, and **move_uploaded_file()** will return `false`. Additionally, a warning will be issued.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

Viz také **is_uploaded_file()**, and the section Handling file uploads for a simple usage example.

pclose (PHP 3, PHP 4)

Closes process file pointer

```
int pclose (int fp)
```

Closes a file pointer to a pipe opened by **popen()**.

The file pointer must be valid, and must have been returned by a successful call to **popen()**.

Returns the termination status of the process that was run.

Viz také **popen()**.

popen (PHP 3, PHP 4)

Opens process file pointer

```
int popen (string command, string mode)
```

Opens a pipe to a process executed by forking the command given by command.

Returns a file pointer identical to that returned by **fopen()**, except that it is unidirectional (may only be used for reading or writing) and must be closed with **pclose()**. This pointer may be used with **fgets()**, **fgetss()**, and **fputs()**.

If an error occurs, returns false.

```
$fp = popen ("/bin/ls", "r");
```

Viz také **pclose()**.

readfile (PHP 3, PHP 4)

Outputs a file

```
int readfile (string filename [, int use_include_path])
```

Reads a file and writes it to standard output.

Returns the number of bytes read from the file. If an error occurs, `false` is returned and unless the function was called as @readfile, an error message is printed.

If *`filename`* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and the text of the response is written to standard output.

Does not handle HTTP redirects, so you must include trailing slashes on directories.

If *`filename`* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and the requested file is written to standard output. If the server does not support passive mode ftp, this will fail.

If *`filename`* begins with neither of these strings, the file will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

Viz také **fpassthru()**, **file()**, **fopen()**, **include()**, **require()**, and **virtual()**.

readlink (PHP 3, PHP 4)

Returns the target of a symbolic link

```
string readlink (string path)
```

Readlink() does the same as the readlink C function and returns the contents of the symbolic link path or 0 in case of error.

Viz také **symlink()**, **readlink()** and **linkinfo()**.

Poznámka: Tato funkce nefunguje na Windows systémech

rename (PHP 3, PHP 4)

Renames a file

```
int rename (string oldname, string newname)
```

Attempts to rename *oldname* to *newname*.

Returns `true` on success and `false` on failure.

rewind (PHP 3, PHP 4)

Rewind the position of a file pointer

```
int rewind (int fp)
```

Sets the file position indicator for fp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**.

Viz také **fseek()** and **ftell()**.

rmdir (PHP 3, PHP 4)

Removes directory

```
int rmdir (string dirname)
```

Attempts to remove the directory named by pathname. The directory must be empty, and the relevant permissions must permit. this.

If an error occurs, returns 0.

Viz také **mkdir()**.

stat (PHP 3, PHP 4)

Gives information about a file

```
array stat (string filename)
```

Gathers the statistics of the file named by filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode

4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type–other systems (i.e. Windows) return -1

The results of this function are cached. See **clearstatcache()** for more details.

lstat (PHP 3>= 3.0.4, PHP 4)

Gives information about a file or symbolic link

```
array lstat (string filename)
```

Gathers the statistics of the file or symbolic link named by filename. This function is identical to the **stat()** function except that if the *filename* parameter is a symbolic link, the status of the symbolic link is returned, not the status of the file pointed to by the symbolic link.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type–other systems (i.e. Windows) return -1

The results of this function are cached. See **clearstatcache()** for more details.

realpath (PHP 4 >= 4.0b4)

Returns canonicalized absolute pathname

```
string realpath (string path)
```

realpath() expands all symbolic links and resolves references to '/./', '/../' and extra '/' characters in the input *path* and return the canonicalized absolute pathname. The resulting path will have no symbolic link, '/./' or '/../' components.

Příklad 1. realpath() example

```
$real_path = realpath ("../../index.php");
```

symlink (PHP 3, PHP 4)

Creates a symbolic link

```
int symlink (string target, string link)
```

symlink() creates a symbolic link from the existing *target* with the specified name *link*.

Viz také **link()** to create hard links, and **readlink()** along with **linkinfo()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech.

tempnam (PHP 3, PHP 4)

Creates unique file name

```
string tempnam (string dir, string prefix)
```

Creates a unique temporary filename in the specified directory. If the directory does not exist, **tempnam()** may generate a filename in the system's temporary directory.

The behaviour of the **tempnam()** function is system dependent. On Windows the TMP environment variable will override the *dir* parameter, on Linux the TMPDIR environment variable has precedence, while SVR4 will always use your *dir* parameter if the directory it points to exists. Consult your system documentation on the tempnam(3) function if in doubt.

Returns the new temporary filename, or the null string on failure.

Příklad 1. Tempnam() example

```
$tmpfname = tempnam ("/tmp", "FOO");
```

> **Poznámka:** This function's behavior changed in 4.0.3. The temporary file is also created to avoid a race condition where the file might appear in the filesystem between the time the string was generated and before the the script gets around to creating the file.

Viz také **tmpfile()**.

tmpfile (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Vytvořit tmp soubor

```
int tmpfile (void)
```

Vytvoří temp soubor, s unikátním názvem, v módu zápisu, a vrací deskriptor tohoto souboru podobně jako **fopen()**. Tento soubor se při zavření (pomocí **fclose()**) nebo při ukončení skriptu automaticky smaže.

Detaily viz systémová dokumentace k funkci `tmpfile(3)` a soubor `stdio.h`.

Viz také **tempnam()**.

touch (PHP 3, PHP 4)

Nastavit čas změny souboru

```
int touch (string filename [, int time])
```

Pokusí se nastavit čas změnu souboru *`filename`* na *`time`*. Pokud *`filename`* není přítomen, použije se aktuální čas.

Pokud tento soubor neexistuje, vytvoří se.

Při úspěchu vrací `true`, jinak `false`.

Příklad 1. Ukázka Touch()

```
if (touch ($FileName)) {
    print "$FileName modification time has been
           changed to todays date and time";
} else {
    print "Sorry Could Not change modification time of $FileName";
}
```

umask (PHP 3, PHP 4)

Změnit současnou umask

```
int umask (int mask)
```

Umask() sets PHP's umask to mask & 0777 and returns the old umask. When PHP is being used as a server module, the umask is restored when each request is finished.

Umask() without arguments simply returns the current umask.

> **Poznámka:** Tato funkce nemusí na Windows fungovat.

unlink (PHP 3, PHP 4)

Smazat soubor

```
int unlink (string filename)
```

Smaže *`filename`*. Podobná Unixové C funkci unlink().

Při chybě vrací 0 nebo FALSE.

Viz také **rmdir()** pro mazání adresářů.

> **Poznámka:** Tato funkce nemusí na Windows fungovat.

XXIV. Forms Data Format functions

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

> **Poznámka:** If you run into problems configuring php with fdftk support, check whether the header file FdfTk.h and the library libFdfTk.so are at the right place. They should be in fdftk-dir/include and fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

The general idea of FDF is similar to HTML forms. The diffence is basically the format how data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (**fdf_create()**) set the values of each input field (**fdf_set_value()**) and associate it with a PDF form (**fdf_set_file()**). Finally it has to be sent to the browser with MimeType `application/vnd.fdf`. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

If you look at an FDF-document with a text editor you will find a catalogue object with the name `FDF`. Such an object may contain a number of entries like `Fields`, `F`, `Status` etc.. The most commonly used entries are `Fields` whicht points to a list of input fields, and `F` which contains the filename of the PDF-document this data belongs to. Those entries are referred to in the FDF documention as /F-Key or /Status-Key. Modifying this entries is done by functions like **fdf_set_file()** and **fdf_set_status()**. Fields are modified with **fdf_set_value()**, **fdf_set_opt()** etc..

The following examples shows just the evaluation of form data.

Příklad 1. Evaluating a FDF document

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
// The pdf form contained several input text fields with the names
// volume, date, comment, publisher, preparer, and two checkboxes
// show_publisher and show_preparer.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volume");
echo "The volume field has the value '<B>$volume</B>'<BR>";

$date = fdf_get_value($fdf, "date");
echo "The date field has the value '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "The publisher field has the value '<B>$publisher</B>'<BR>";
} else
  echo "Publisher shall not be shown.<BR>";

if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "The preparer field has the value '<B>$preparer</B>'<BR>";
} else
  echo "Preparer shall not be shown.<BR>";
fdf_close($fdf);
?>
```

fdf_open (PHP 3>= 3.0.6, PHP 4)

Open a FDF document

```
int fdf_open (string filename)
```

The **fdf_open()** function opens a file with form data. This file must contain the data as returned from a PDF form. Currently, the file has to be created 'manually' by using **fopen()** and writing the content of HTTP_FDF_DATA with **fwrite()** into it. A mechanism like for HTML form data where for each input field a variable is created does not exist.

Příklad 1. Accessing the form data

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

See also **fdf_close()**.

fdf_close (PHP 3>= 3.0.6, PHP 4)

Close an FDF document

```
boolean fdf_close (int fdf_document)
```

The **fdf_close()** function closes the FDF document.

See also **fdf_open()**.

fdf_create (PHP 3>= 3.0.6, PHP 4)

Create a new FDF document

```
int fdf_create (void )
```

The **fdf_create()** creates a new FDF document. This function is needed if one would like to populate input fields in a PDF document with data.

Příklad 1. Populating a PDF document

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
```

```
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

See also **fdf_close()**, **fdf_save()**, **fdf_open()**.

fdf_save (PHP 3>= 3.0.6, PHP 4)

Save a FDF document

```
int fdf_save (string filename)
```

The **fdf_save()** function saves a FDF document. The FDF Toolkit provides a way to output the document to stdout if the parameter *filename* is '.'. This does not work if PHP is used as an apache module. In such a case one will have to write to a file and use e.g. **fpassthru()**. to output it.

See also **fdf_close()** and example for **fdf_create()**.

fdf_get_value (PHP 3>= 3.0.6, PHP 4)

Get the value of a field

```
string fdf_get_value (int fdf_document, string fieldname)
```

The **fdf_get_value()** function returns the value of a field.

See also **fdf_set_value()**.

fdf_set_value (PHP 3>= 3.0.6, PHP 4)

Set the value of a field

```
boolean fdf_set_value (int fdf_document, string fieldname, string value, int isName)
```

The **fdf_set_value()** function sets the value of a field. The last parameter determines if the field value is to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0).

See also **fdf_get_value()**.

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4)

Get the next field name

```
string fdf_next_field_name (int fdf_document, string fieldname)
```

The **fdf_next_field_name()** function returns the name of the field after the field in *fieldname* or the field name of the first field if the second paramter is NULL.

See also **fdf_set_field()**, **fdf_get_field()**.

fdf_set_ap (PHP 3>= 3.0.6, PHP 4)

Set the appearance of a field

```
boolean fdf_set_ap (int fdf_document, string field_name, int face, string filename, int page_number)
```

The **fdf_set_ap()** function sets the appearance of a field (i.e. the value of the /AP key). The possible values of *face* are 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status (PHP 3>= 3.0.6, PHP 4)

Set the value of the /STATUS key

```
boolean fdf_set_status (int fdf_document, string status)
```

The **fdf_set_status()** sets the value of the /STATUS key.

See also **fdf_get_status()**.

fdf_get_status (PHP 3>= 3.0.6, PHP 4)

Get the value of the /STATUS key

```
string fdf_get_status (int fdf_document)
```

The **fdf_get_status()** returns the value of the /STATUS key.

See also **fdf_set_status()**.

fdf_set_file (PHP 3>= 3.0.6, PHP 4)

Set the value of the /F key

```
boolean fdf_set_file (int fdf_document, string filename)
```

The **fdf_set_file()** sets the value of the /F key. The /F key is just a reference to a PDF form which is to be populated with data. In a web environment it is a URL (e.g. http:/testfdf/resultlabel.pdf).

See also **fdf_get_file()** and example for **fdf_create()**.

fdf_get_file (PHP 3>= 3.0.6, PHP 4)

Get the value of the /F key

```
string fdf_get_file (int fdf_document)
```

The **fdf_set_file()** returns the value of the /F key.

See also **fdf_set_file()**.

fdf_set_flags (PHP 4 >= 4.0.2)

Sets a flag of a field

```
boolean fdf_set_flags (int fdf_document, string fieldname, int whichFlags, int newFlags)
```

The **fdf_set_flags()** sets certain flags of the given field *fieldname*.

See also **fdf_set_opt()**.

fdf_set_opt (PHP 4 >= 4.0.2)

Sets an option of a field

```
boolean fdf_set_opt (int fdf_document, string fieldname, int element, string str1,
string str2)
```

The **fdf_set_opt()** sets options of the given field *fieldname*.

See also **fdf_set_flags()**.

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Sets an javascript action of a field

```
boolean fdf_set_submit_form_action (int fdf_document, string fieldname, int trigger,
string script, int flags)
```

The **fdf_set_submit_form_action()** sets a submit form action for the given field *fieldname*.

See also **fdf_set_javascript_action()**.

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Sets an javascript action of a field

```
boolean fdf_set_javascript_action (int fdf_document, string fieldname, int trigger,
string script)
```

The **fdf_set_javascript_action()** sets a javascript action for the given field *fieldname*.

See also **fdf_set_submit_form_action()**.

XXV. FTP functions

FTP stands for File Transfer Protocol.

The following constants are defined when using the FTP module: `FTP_ASCII` and `FTP_BINARY`.

Příklad 1. ftp() example

```
<?php
// set up basic connection
$conn_id = ftp_connect("$ftp_server");

// login with username and password
$login_result = ftp_login($conn_id, "$ftp_user_name", "$ftp_user_pass");

// check connection
if ((!$conn_id) || (!$login_result)) {
        echo "Ftp connection has failed!";
        echo "Attempted to connect to $ftp_server for user $user";
        die;
    } else {
        echo "Connected to $ftp_server, for user $user";
    }

// upload the file
$upload = ftp_put($conn_id, "$destination_file", "$source_file", FTP_BINARY);

// check upload status
if (!$upload) {
        echo "Ftp upload has failed!";
    } else {
        echo "Uploaded $source_file to $ftp_server as $destination_file";
    }

// close the FTP stream
ftp_quit($conn_id);
?>
```

ftp_connect (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Opens up an FTP connection

```
int ftp_connect (string host [, int port])
```

Returns a FTP stream on success, false on error.

ftp_connect() opens up a FTP connection to the specified *host*. The *port* parameter specifies an alternate port to connect to. If it is omitted or zero, then the default FTP port, 21, will be used.

ftp_login (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Logs in an FTP connection

```
int ftp_login (int ftp_stream, string username, string password)
```

Returns true on success, false on error.

Logs in the given FTP stream.

ftp_pwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the current directory name

```
string ftp_pwd (int ftp_stream)
```

Returns the current directory, or false on error.

ftp_cdup (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Changes to the parent directory

```
int ftp_cdup (int ftp_stream)
```

Returns true on success, false on error.

Changes to the parent directory.

ftp_chdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Changes directories on a FTP server

```
int ftp_chdir (int ftp_stream, string directory)
```

Returns true on success, false on error.

Changes to the specified *directory*.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Creates a directory

```
string ftp_mkdir (int ftp_stream, string directory)
```

Returns the newly created directory name on success, false on error.

Creates the specified *directory*.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Removes a directory

```
int ftp_rmdir (int ftp_stream, string directory)
```

Returns true on success, false on error.

Removes the specified *directory*.

ftp_nlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns a list of files in the given directory.

```
array ftp_nlist (int ftp_stream, string directory)
```

Returns an array of filenames on success, false on error.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns a detailed list of files in the given directory.

```
array ftp_rawlist (int ftp_stream, string directory)
```

ftp_rawlist() executes the FTP LIST command, and returns the result as an array. Each array element corresponds to one line of text. The output is not parsed in any way. The system type identifier returned by **ftp_systype()** can be used to determine how the results should be interpreted.

ftp_systype (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the system type identifier of the remote FTP server.

```
string ftp_systype (int ftp_stream)
```

Returns the remote system type, or false on error.

ftp_pasv (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Turns passive mode on or off.

```
int ftp_pasv (int ftp_stream, int pasv)
```

Returns true on success, false on error.

ftp_pasv() turns on passive mode if the *pasv* parameter is true (it turns off passive mode if *pasv* is false.) In passive mode, data connections are initiated by the client, rather than by the server.

ftp_get (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Downloads a file from the FTP server.

```
int ftp_get (int ftp_stream, string local_file, string remote_file, int mode)
```

Returns true on success, false on error.

ftp_get() retrieves *remote_file* from the FTP server, and saves it to *local_file* locally. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY.

ftp_fget (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Downloads a file from the FTP server and saves to an open file.

```
int ftp_fget (int ftp_stream, int fp, string remote_file, int mode)
```

Returns true on success, false on error.

ftp_fget() retrieves *remote_file* from the FTP server, and writes it to the given file pointer, *fp*. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY.

ftp_put (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Uploads a file to the FTP server.

```
int ftp_put (int ftp_stream, string remote_file, string local_file, int mode)
```

Returns true on success, false on error.

ftp_put() stores *local_file* on the FTP server, as *remote_file*. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY.

Příklad 1. Ftp_put() example

```
$upload = ftp_put ($conn_id, "$destination_file", "$source_file", FTP_ASCII);
```

ftp_fput (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Uploads from an open file to the FTP server.

```
int ftp_fput (int ftp_stream, string remote_file, int fp, int mode)
```

Returns true on success, false on error.

ftp_fput() uploads the data from the file pointer *fp* until end of file. The results are stored in *remote_file* on the FTP server. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY

ftp_size (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the size of the given file.

```
int ftp_size (int ftp_stream, string remote_file)
```

Returns the file size on success, or -1 on error.

ftp_size() returns the size of a file. If an error occurs, of if the file does not exist, -1 is returned. Not all servers support this feature.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the last modified time of the given file.

```
int ftp_mdtm (int ftp_stream, string remote_file)
```

Returns a UNIX timestamp on success, or -1 on error.

ftp_mdtm() checks the last-modified time for a file, and returns it as a UNIX timestamp. If an error occurs, or the file does not exist, -1 is returned. Note that not all servers support this feature.

> **Poznámka: ftp_mdtm()** does not work with directories.

ftp_rename (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Renames a file on the ftp server.

```
int ftp_rename (int ftp_stream, string from, string to)
```

Returns true on success, false on error.

ftp_rename() renames the file specified by *from* to the new name *to*

ftp_delete (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Deletes a file on the ftp server.

```
int ftp_delete (int ftp_stream, string path)
```

Returns true on success, false on error.

ftp_delete() deletes the file specified by *path* from the FTP server.

ftp_site (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Sends a SITE command to the server.

```
int ftp_site (int ftp_stream, string cmd)
```

Returns true on success, false on error.

ftp_site() sends the command specified by *cmd* to the FTP server. SITE commands are not standardized, and vary from server to server. They are useful for handling such things as file permissions and group membership.

ftp_quit (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Closes an FTP connection

```
int ftp_quit (int ftp_stream)
```

ftp_connect() closes *ftp_stream*.

XXVI. Funkce pro práci s funkcemi

Všechny tyto funkce pokrývají různé aspekty práce s funkcemi.

call_user_func (PHP 3>= 3.0.3, PHP 4)

Zavolat uživatelskou funkci určenou prvním argumentem

```
mixed call_user_func  (string function_name [, mixed parameter [, mixed ...]])
```

Zavolá uživatelsky definouvanou funkci určenou argumentem *function_name*. Zvažte následující ukázku:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func (’barber’, "mushroom");
call_user_func (’barber’, "shave");
```

create_function (PHP 4 >= 4.0.1)

Vytvořit anonymní (lambda-style) funkci

```
string create_function (string args, string code)
```

Z předaných argumentů vytvoří anonymní funkci, a vrátí unikátní název této funkce. *args* se obvykle předává jako string v jednoduchých uvozovkách, a doporučujeme to i pro *code*. Důvodem pro jednoduché uvozovky je ochrana názvů proměnných před parsováním, pokud použijete dvojité uvozovky, budete muset oescapovat názvy proměnných, např. `\$avar`.

Tuto funkci můžete (například) použít k vytvoření funkce z dat shromážděných za běhu programu:

Příklad 1. Vytvoření anonymní funkce pomocí create_function()

```
$newfunc = create_function(’$a,$b’,’return "ln($a) + ln($b) = ".log($a * $b);’);
echo "New anonymous function: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// výstup
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Nebo třeba k vytvoření obecného handleru, který na předané argumenty aplikuje sadu operací:

Příklad 2. Vytvoření obecné zpracující funkce pomocí create_function()

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
    echo $farr[$f]($var1,$var2)."\n";
}

// create a bunch of math functions
$f1 = ’if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}’;
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = ’if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return false;}’;
$farr = array(
    create_function(’$x,$y’, ’return "some trig: ".(sin($x) + $x*cos($y));’),
    create_function(’$x,$y’, ’return "a hypotenuse: ".sqrt($x*$x + $y*$y);’),
    create_function(’$a,$b’, $f1),
    create_function(’$a,$b’, $f2),
    create_function(’$a,$b’, $f3)
    );

echo "\nUsing the first array of anonymous functions\n";
```

```
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
    'and \"$b\"\n** Look the same to me! (looking at the first 3 chars)";'),
    create_function('$a,$b,'; return "CRCs: ".crc32($a)." , ".crc32(b);'),
    create_function('$a,$b,'; return "simi-
lar(a,b) = ".similar_text($a,$b,&$p)."($p%)";')
    );
echo "\nUsing the second array of anonymous functions\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
```

když spustíte výšeuvedený kód, výstup bude:

```
Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

Ale zřejmě nejběžnějším využitím lambda-style (anonymních) funkcí je tvorba callback funkcí, např. pro použití v **array_walk()** nebo **usort()**

Příklad 3. Využití anonymních funkcí jako callback funkcí

```
$av = array("the ","a ","that ","this ");
array_walk($av, create_function('&$v,$k','$v = $v."mango";'));
print_r($av);  // for PHP 3 use var_dump()
// outputs:
// Array
// (
//   [0] => the mango
//   [1] => a mango
//   [2] => that mango
//   [3] => this mango
// )

// an array of strings ordered from shorter to longer
$sv = array("small","larger","a big string","it is a string thing");
print_r($sv);
// outputs:
// Array
// (
//   [0] => small
//   [1] => larger
//   [2] => a big string
//   [3] => it is a string thing
// )

// sort it from longer to shorter
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
```

```
//   [0] => it is a string thing
//   [1] => a big string
//   [2] => larger
//   [3] => small
// )
```

func_get_arg (PHP 4 >= 4.0b4)

Vrátit položku ze seznamu argumentů

```
mixed func_get_arg (int arg_num)
```

Vrací argument, který je na *arg_num*-té pozici v seznamu argumentů užiavtelsky definované funkce. Argumenty funkcí se počítají od nuly. **func_get_arg()** při volání mimo definici funkce generuje varování.

Pokud je *arg_num* větší než počet skutečně předaných argumentů, vygeneruje varování a vrátí `false`.

```
<?php
function foo() {
     $numargs = func_num_args();
     echo "Number of arguments: $numargs<br>\n";
     if ($numargs >= 2) {
     echo "Second argument is: " . func_get_arg (1) . "<br>\n";
     }
}

foo (1, 2, 3);
?>
```

func_get_arg() se dá použít ve spojení s **func_num_args()** a **func_get_args()** k tvorbě uživatelsky definovaných funkcí, které přijímají proměnný počet argumentů.

> **Poznámka:** Tato funkce byla přidána v PHP 4.

func_get_args (PHP 4 >= 4.0b4)

Vrátit pole obsahující seznam argumentů funkce

```
array func_get_args (void )
```

Vrací pole, jehož každý prvek je odpovídající položkou seznamu argumentů současné uživatelsky definované funkce. **func_get_args()** při volání mimo definici funkce generuje varování.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br>\n";
    if ($numargs >= 2) {
    echo "Second argument is: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
```

```
    echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

func_get_args() se dá použít ve spojení s **func_num_args()** a **func_get_arg()** k tvorbě uživatelsky definovaných funkcí, které přijímají proměnny počet argumentů.

> **Poznámka:** Tato funkce byla přidána v PHP 4.

func_num_args (PHP 4 >= 4.0b4)

Vrátit počet argumentů předaných funkci

```
int func_num_args (void )
```

Vrací počet argumentů předaných současné uživatelsky definované funkci. **Func_num_args()** pri volání mimo definici funkce generuje warning. definition.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs\n";
}

foo (1, 2, 3);    // Prints 'Number of arguments: 3'
?>
```

func_num_args() se dá použit ve spojení s **func_get_arg()** a **func_get_args()** k tvorbě uživatelsky definovaných funkcí, které přijímají proměnný počet argumentů.

> **Poznámka:** Tato funkce byla přidána v PHP 4.

function_exists (PHP 3>= 3.0.7, PHP 4)

Vrátit true, pokud je daná funkce definována

```
bool function_exists (string function_name)
```

Ověří přítomnost funkce *`function_name`* v seznamu definovaných funkcí. Pokud daná funkce existuje, vrací `true`, jinak `false`.

```
if (function_exists('imap_open')) {
    echo "IMAP functions are available.<br>\n";
} else {
    echo "IMAP functions are not available.<br>\n";
}
```

Pozn.: název funkce může existovat i v případě, že je samotná funkce nepoužitelná kvůli konfiguraci nebo volbám zadaným při kompilaci.

Viz také **method_exists()**.

register_shutdown_function (PHP 3>= 3.0.4, PHP 4)

Zaregistrovat funkci pro provedení při ukončení běhu

```
int register_shutdown_function  (string func)
```

Zaregistruje funkci udanou v *func* pro provedení po dokončení běhu skriptu.

Běžné problémy:

Jelikož tato funkce nemůže poslat browseru žádný výstup, nebudete ji moci debugovat pomocí příkazů jako **print()** nebo **echo()**.

XXVII. GNU Gettext

Gettext funkce implementujíc NLS (Native Language Support) API, která se dá použít k internacionalizaci vašich PHP aplikací. Důkladné vysvětlení těchto funkcí viz dokumentaci GNU Gettext.

bindtextdomain (PHP 3>= 3.0.7, PHP 4)

Nastavit cestu pro doménu

```
string bindtextdomain (string domain, string directory)
```

Funkce **bindtextdomain()** nastaví cestu pro doménu.

dcgettext (PHP 3>= 3.0.7, PHP 4)

Změnit doménu pro jediné vyhledání

```
string dcgettext (string domain, string message, int category)
```

Tato funkce vám umožní zmenit současnou doménu pro jediné vyhledání zprávy. Umožňuje také specifikovat kategorii.

dgettext (PHP 3>= 3.0.7, PHP 4)

Změnit současnou doménu

```
string dgettext (string domain, string message)
```

Funkce **dgettext()** umožňuje změnit současnou doménu pro jediné vyhledání zprávy.

gettext (PHP 3>= 3.0.7, PHP 4)

Vyhledat zprávu v současné doméně

```
string gettext (string message)
```

Tato funkce vrátí přeložený řetězec, pokud jej najde v překladové tabulce, nebo předaný řetězec, pokud jej nenajde. Jako alias k této funkci můžete použít podtržítko.

Příklad 1. Gettext()-check

```
<?php
// Set language to German
putenv ("LANG=de");

// Specify location of translation tables
bindtextdomain ("myPHPApp", "./locale");

// Choose domain
textdomain ("myPHPApp");

// Print a test message
print (gettext ("Vítejte v mé PHP Aplikaci"));
?>
```

textdomain (PHP 3>= 3.0.7, PHP 4)

Nastavit výchozí doménu

```
string textdomain (string text_domain)
```

Tato funkce nastaví doménu pro vyhledávání při volání funkce **gettext()**, obvykle pojmenovanou podle aplikace. Vrátí předchozí výchozí doménu. Při volání bez argumentů vrátí současné nastavení, aniž by jej měnila.

XXVIII. GMP functions

These functions allow you to work with arbitrary-length integers using the GNU MP library. In order to have these functions available, you must compile PHP with GMP support by using the `-with-gmp` option.

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

These functions have been added in PHP 4.0.4.

> **Poznámka:** Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the **gmp_init()** function.

Příklad 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factiorial of 1000 (pretty big number) very fast.

gmp_init (PHP 4 >= 4.0.4)

Create GMP number

```
resource gmp_init (mixed number)
```

Creates a GMP number from an integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with `0x`.

Příklad 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Poznámka: It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like **gmp_add()**. Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convert GMP number to integer

```
int gmp_intval (resource gmpnumber)
```

This function allows to convert GMP number to integer.

Varování

This function returns a useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use **gmp_strval()**.

gmp_strval (PHP 4 >= 4.0.4)

Convert GMP number to string

```
string gmp_strval (resource gmpnumber [, int base])
```

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Příklad 1. Converting a GMP number to a string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_add (PHP 4 >= 4.0.4)

Add numbers

```
resource gmp_add (resource a, resource b)
```

Add two GMP numbers. The result will be a GMP number representing the sum of the arguments.

gmp_sub (PHP 4 >= 4.0.4)

Subtract numbers

```
resource gmp_sub (resource a, resource b)
```

Subtracts *b* from *a* and returns the result.

gmp_mul (PHP 4 >= 4.0.4)

Multiply numbers

```
resource gmp_mul (resource a, resource b)
```

Multiplies *a* by *b* and returns the result.

gmp_div_q (PHP 4 >= 4.0.4)

Divide numbers

```
resource gmp_div_q (resource a, resource b [, int round])
```

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards `+infinity`.
- *GMP_ROUND_MINUSINF*: The result is rounded towards `-infinity`.

This function can also be called as **gmp_div()**.

See also **gmp_div_r()**, **gmp_div_qr()**

gmp_div_r (PHP 4 >= 4.0.4)

Remainder of the division of numbers

```
resource gmp_div_r (resource n, resource d [, int round])
```

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the **gmp_div_q()** function for description of the *round* argument.

See also **gmp_div_q()**, **gmp_div_qr()**

gmp_div_qr (PHP 4 >= 4.0.4)

Divide numbers and get quotient and remainder

```
array gmp_div_qr (resource n, resource d [, int round])
```

The function divides *n* by *d* and returns array, with the first element being `[n/d]` (the integer result of the division) and the second being `(n - [n/d] * d)` (the remainder of the division).

See the **gmp_div_q()** function for description of the *round* argument.

Příklad 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also **gmp_div_q()**, **gmp_div_r()**.

gmp_div (PHP 4 >= 4.0.4)

Divide numbers

```
resource gmp_div (resource a, resource b)
```

This function is an alias to **gmp_div_q()**.

gmp_mod (PHP 4 >= 4.0.4)

Modulo operation

```
resource gmp_mod (resource n, resource d)
```

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_divexact (PHP 4 >= 4.0.4)

Exact division of numbers

```
resource gmp_divexact (resource n, resource d)
```

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_cmp (PHP 4 >= 4.0.4)

Compare numbers

```
int gmp_cmp (resource a, resource b)
```

Returns a positive value if `a > b`, zero if `a = b` and negative value if `a < b`.

gmp_neg (PHP 4 >= 4.0.4)

Negate number

```
resource gmp_neg (resource a)
```

Returns -*a*.

gmp_abs (PHP 4 >= 4.0.4)

Absolute value

```
resource gmp_abs (resource a)
```

Returns absolute value of *a*.

gmp_sign (PHP 4 >= 4.0.4)

Sign of number

```
int gmp_sign (resource a)
```

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_fact (PHP 4 >= 4.0.4)

Factorial

```
resource gmp_fact (int a)
```

Calculates factorial (`a!`) of *a*.

gmp_sqrt (PHP 4 >= 4.0.4)

Square root

```
resource gmp_sqrt (resource a)
```

Calculates square root of *a*.

gmp_sqrtrm (unknown)

Square root with remainder

```
array gmp_sqrtrm (resource a)
```

Returns array where first element is the integer square root of *a* (see also **gmp_sqrt()**), and the second is the remainder (i.e., the difference between *a* and the first element squared).

gmp_perfect_square (PHP 4 >= 4.0.4)

Perfect square check

```
bool gmp_perfect_square (resource a)
```

Returns true if *a* is a perfect square, false otherwise.

See also: **gmp_sqrt()**, **gmp_sqrtrm()**.

gmp_pow (PHP 4 >= 4.0.4)

Raise number into power

```
resource gmp_pow (resource base, int exp)
```

Raise *base* into power *exp*. The case of 0^0 yields 1. *exp* cannot be negative.

gmp_powm (PHP 4 >= 4.0.4)

Raise number into power with modulo

```
resource gmp_powm (resource base, resource exp, resource mod)
```

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime (PHP 4 >= 4.0.4)

Check if number is "probably prime"

```
int gmp_prob_prime (resource a [, int reps])
```

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin’s probabilistic test.

gmp_gcd (PHP 4 >= 4.0.4)

Calculate GCD

```
resource gmp_gcd (resource a, resource b)
```

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of, or both, input operands are negative.

gmp_gcdext (PHP 4 >= 4.0.4)

Calculate GCD and multipliers

```
array gmp_gcdext (resource a, resource b)
```

Calculates g, s, and t, such that `a*s + b*t = g = gcd(a,b)`, where gcd is the greatest common divisor. Returns an array with respective elements g, s and t.

gmp_invert (PHP 4 >= 4.0.4)

Inverse by modulo

```
resource gmp_invert (resource a, resource b)
```

Computes the inverse of *a* modulo *b*. Returns false if an inverse does not exist.

gmp_legendre (PHP 4 >= 4.0.4)

Legendre symbol

```
int gmp_legendre (resource a, resource p)
```

Compute the Legendre symbol (http://www.utm.edu/research/primes/glossary/LegendreSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_jacobi (PHP 4 >= 4.0.4)

Jacobi symbol

```
int gmp_jacobi (resource a, resource p)
```

Computes Jacobi symbol (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_random (PHP 4 >= 4.0.4)

Random number

```
resource gmp_random (int limiter)
```

Generate a random number. The number will be up to *limiter* words long. If *limiter* is negative, negative numbers are generated.

gmp_and (PHP 4 >= 4.0.4)

Logical AND

```
resource gmp_and (resource a, resource b)
```

Calculates logical AND of two GMP numbers.

gmp_or (PHP 4 >= 4.0.4)

Logical OR

```
resource gmp_or (resource a, resource b)
```

Calculates logical inclusive OR of two GMP numbers.

gmp_xor (PHP 4 >= 4.0.4)

Logical XOR

```
resource gmp_xor (resource a, resource b)
```

Calculates logical exclusive OR (XOR) of two GMP numbers.

gmp_setbit (PHP 4 >= 4.0.4)

Set bit

```
resource gmp_setbit (resource &a, int index [, bool set_clear])
```

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_clrbit (PHP 4 >= 4.0.4)

Clear bit

```
resource gmp_clrbit (resource &a, int index)
```

Clears (sets to 0) bit *index* in *a*.

gmp_scan0 (PHP 4 >= 4.0.4)

Scan for 0

```
int gmp_scan0 (resource a, int start)
```

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1 (PHP 4 >= 4.0.4)

Scan for 1

```
int gmp_scan1 (resource a, int start)
```

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_popcount (PHP 4 >= 4.0.4)

Population count

```
int gmp_popcount (resource a)
```

Return the population count of *a*.

gmp_hamdist (PHP 4 >= 4.0.4)

Hamming distance

```
int gmp_hamdist (resource a, resource b)
```

Returns the hamming distance between *a* and *b*. Both operands should be non-negative.

XXIX. HTTP funkce

Tyto funkce vám umožňují manipulovat výstupem posílaným zpět browseru přímo na úrovni HTTP protokolu.

header (PHP 3, PHP 4)

Poslat HTTP hlavičku

```
int header (string string)
```

Funkce **Header()** se používá na začátku HTML souboru k odeslání HTTP hlaviček. Více informací o HTTP hlavičkách viz Specifikace HTTP 1.1 (http://www.w3.org/Protocols/rfc2616/rfc2616). *Poznámka:* Pamatujte, že funkce **Header()** musí být volána dříve než se odešle jakýkoliv normální výstup, ať už normálními HTML tagy, nebo z PHP. Velmi obvyklou chybou je načítat kód pomocí **include()** nebo auto_prepend a mít v tomto kódu prázdné řádky, které způsobí odeslání výstupu před voláním funkce **header()**.

Existují dva zvláštní případy volání funkce **header()**. Prvním je hlavička "Location". Ta nejenže odešle hlavičku browseru, ale navíc i vrátí Apachi stavový kód REDIRECT. Z pohledu autora skriptu by to nemělo být důležité, ale je to důležité pro lidi, kteří rozumí vnitřnostem Apache.

```
header ("Location: http://www.php.net"); /* Přesměrujeme browser
                                            na web site PHP */
exit;                 /* Pojistíme si, že se další kód nevykoná po
                         přesměrování. */
```

Druhým zvláštním případem jsou všechny hlavičky začínající řetězcem "HTTP/" (velikost písmen nehraje roli). Například, pokud direktiva ErrorDocument 404 vašeho Apache ukazuje na PHP skript, nebylo by od věci, kdyby skutečně generoval 404. První věcí, kterou byste v tomto skriptu měli udělat tudíz bude:

```
header ("HTTP/1.0 404 Not Found");
```

PHP skripty často generují dynamické HTML, které nesmí být cachováno uživatelským browserem, ani žýdnými proxynami mezi serverem a uživatelským browserem. Mnoho proxyn a klientů se dá donutit k vypnutí cachování s pomocí

```
header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // datum v minulosti
header ("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
                                                      // vždy upraven
header ("Cache-Control: no-cache, must-revalidate");  // HTTP/1.1
header ("Pragma: no-cache");                          // HTTP/1.0
```

Viz také **headers_sent()**

headers_sent (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Vrátit true, pokud byly odeslány hlavičky

```
boolean headers_sent (void)
```

Tato funkce vrátí `true`, pokud už byly HTTP hlavičky odeslány, jinak `false`.

Viz také **header()**

setcookie (PHP 3, PHP 4)

Poslat cookie

```
int setcookie (string name [, string value [, int expire [, string path [, string domain
[, int secure]]]]])
```

Setcookie() definuje cookie, který se pošle spolu s hlavičkami. Cookies se musí odeslat jako *první* ze všech HTTP hlaviček (to je omezení cookies, ne PHP). Volání této funkce musíte tudíž umístit před `<html>` či `<head>` tagy.

Všechny argumenty kromě argumentu `name` jsou nepovinné. Pokud je přítomný pouze argument `name`, u klienta se smaže cookie tohoto jména. Kterýkoliv argument můžete také nahradit prázdným řetězcem (`""`), čímž tento argument přeskočíte. Argumenty `expire` a `secure` jsou celočíselné a nedají se přeskočit prázdným řetězcem. Místo toho použijte nulu (*0*). Argument `expire` je běžné Unixové celočíselné vyjádření času, jak je vrací funkce **time()** či **mktime()**. Argument `secure` indikuje, že by se tento cookie měl přenášet pouze po zabezpečeném HTTPS spojení.

Běžné zádrhele:

- Cookies jsou přístupné až při dalším načtení stránky, na které přístupné být mají.
- Cookies se musí mazat se stejnými parametry, se kterými byly odeslány.

V PHP 3 se vícenásobná volání **setcookie()** v jednom skriptu provedou v opačném pořadí. Pokud se pokoušíte smazat jeden cookie pře odesláním jiného, měli byste umístit vložení před smazání. V PHP 4 se vícenásobná volání **setcookie()** provedou v tom pořadí, jak jsou volána.

Několik příkladů, jak posílat cookies:

Příklad 1. Ukázky odeslání cookies pomocí setcookie()

```
setcookie ("TestCookie", "Zkušební hodnota");
setcookie ("TestCookie", $value,time()+3600);  /* vyprší za hodinu */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".utoronto.ca", 1);
```

Následují příklady mazání cookies z předchozí ukázky:

Příklad 2. Ukázky mazání cookies pomocí setcookie()

```
setcookie ("TestCookie");
// nastaví dobu vypršení na čas před hodinou
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".utoronto.ca", 1);
```

Při mazání cookie byste se měli ujistit, že je doba vypršení v minulosti, čímž se v browseru zapne mechanismus odstranění cookie.

Všimněte si, že hodnotová část cookie se při odeslání cookie automaticky url-zakóduje, a při přijetí se automaticky dekóduje a přiřadí proměnné stejného jména, jako je jméno cookie. Pokud chcete vidět obsah našeho zkušebního cookie, použijte některý z následujících příkladů:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

Cookies obsahující pole můžete také odeslat tak, že za název cookie napíšete hranaté závorky. Účinkem tohoto je odeslání tolika cookies, kolik má vaše pole prvků, ale když váš skript tyto cookies přijme, hodnoty se umístí v poli stejného jména, jako jsou vaše cookies:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
    while (list ($name, $value) = each ($cookie)) {
        echo "$name == $value<br>\n";
```

```
    }
}
```

Více informací o cookies viz specifikace cookies na http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 se Service Packem 1 zpracovává chybně cookies, které mají nastavený argument `path`.

Netscape Communicator 4.05 a Microsoft Internet Explorer 3.x zřejmě nezpracují cookies správně, pokud nejsou nastaveny argumenty `path` a `time`.

XXX. Hyperwave functions

Introduction

Hyperwave has been developed at IICM (http://iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 4.1, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user. An attribute is a name/value pair of the form name=value. The complete object record contains as many of those pairs as the user likes. The name of an attribute does not have to be unique, e.g. a title may appear several times within an object record. This makes sense if you want to specify a title in several languages. In such a case there is a convention, that each title value is preceded by the two letter language abbreviation followed by a colon, e.g. 'en:Title in English' or 'ge:Titel in deutsch'. Other attributes like a description or keywords are potential candidates. You may also replace the language abbreviation by any other string as long as it separated by colon from the rest of the attribute value.

Each object record has native a string representation with each name/value pair separated by a newline. The Hyperwave extension also knows a second representation which is an associated array with the attribute name being the key. Multilingual attribute values itself form another associated array with the key being the language abbreviation. Actually any multiple attribute forms an associated array with the string left to the colon in the attribute value being the key. (This is not fully implemented. Only the attributes Title, Description and Keyword are treated properly yet.)

Besides the documents, all hyper links contained in a document are stored as object records as well. Hyper links which are in a document will be removed from it and stored as individual objects, when the document is inserted into the database. The object record of the link contains information about where it starts and where it ends. In order to gain the original document you will have to retrieve the plain document without the links and the list of links and reinsert them (The functions **hw_pipedocument()** and **hw_gettext()** do this for you. The advantage of separating links from the document is obvious. Once a document to which a link is pointing to changes its name, the link can easily be modified accordingly. The document containing the link is not affected at all. You may even add a link to a document without modifying the document itself.

Saying that **hw_pipedocument()** and **hw_gettext()** do the link insertion automatically is not as simple as it sounds. Inserting links implies a certain hierarchy of the documents. On a web server this is given by the file system, but Hyperwave has its own hierarchy and names do not reflect the position of an object in that hierarchy. Therefore creation of links first of all requires a mapping from the Hyperwave hierarchy and namespace into a web hierarchy respective web namespace. The fundamental difference between Hyperwave and the web is the clear distinction between names and hierarchy in Hyperwave. The name does not contain any information about the objects position in the hierarchy. In the web the name also contains the information on where the object is located in the hierarchy. This leads to two possibles ways of mapping. Either the Hyperwave hierarchy and name of the Hyperwave object is reflected in the URL or the name only. To make things simple the second approach is used. Hyperwave object with name 'my_object' is mapped to 'http://host/my_object' disregarding where it resides in the Hyperwave hierarchy. An object with name 'parent/my_object' could be the child of 'my_object' in the Hyperwave hierarchy, though in a web namespace it appears to be just the opposite and the user might get confused. This can only be prevented by selecting reasonable object names.

Having made this decision a second problem arises. How do you involve PHP? The URL http://host/my_object will not call any PHP script unless you tell your web server to rewrite it to e.g. 'http://host/php3_script/my_object' and the script 'php3_script' evaluates the $PATH_INFO variable and retrieves the object with name 'my_object' from the Hyperwave server. Their is just one little drawback which can be fixed easily. Rewriting any URL would not allow any access to other document on the web server. A PHP script for searching in the Hyperwave server would be impossible. Therefore you will need at least a second rewriting rule to exclude certain URLS like all e.g. starting with http://host/Hyperwave. This is basically sharing of a namespace by the web and Hyperwave server.

Based on the above mechanism links are insert into documents.

It gets more complicated if PHP is not run as a server module or CGI script but as a standalone application e.g. to dump the content of the Hyperwave server on a CD-ROM. In such a case it makes sense to retain the Hyperwave

hierarchy and map in onto the file system. This conflicts with the object names if they reflect its own hierarchy (e.g. by choosing names including '/'). Therefore '/' has to be replaced by another character, e.g. '_'. to be continued.

The network protocol to communicate with the Hyperwave server is called HG-CSP (http://www.hyperwave.com/7.17-hg-prot) (Hyper-G Client/Server Protocol). It is based on messages to initiate certain actions, e.g. get object record. In early versions of the Hyperwave Server two native clients (Harmony, Amadeus) were provided for communication with the server. Those two disappeared when Hyperwave was commercialised. As a replacement a so called wavemaster was provided. The wavemaster is like a protocol converter from HTTP to HG-CSP. The idea is to do all the administration of the database and visualisation of documents by a web interface. The wavemaster implements a set of placeholders for certain actions to customise the interface. This set of placeholders is called the PLACE Language. PLACE lacks a lot of features of a real programming language and any extension to it only enlarges the list of placeholders. This has led to the use of JavaScript which IMO does not make life easier.

Adding Hyperwave support to PHP should fill in the gap of a missing programming language for interface customisation. It implements all the messages as defined by the HG-CSP but also provides more powerful commands to e.g. retrieve complete documents.

Hyperwave has its own terminology to name certain pieces of information. This has widely been taken over and extended. Almost all functions operate on one of the following data types.

- object ID: An unique integer value for each object in the Hyperwave server. It is also one of the attributes of the object record (ObjectID). Object ids are often used as an input parameter to specify an object.
- object record: A string with attribute-value pairs of the form attribute=value. The pairs are separated by a carriage return from each other. An object record can easily be converted into an object array with **hw_object2array()**. Several functions return object records. The names of those functions end with obj.
- object array: An associated array with all attributes of an object. The key is the attribute name. If an attribute occurs more than once in an object record it will result in another indexed or associated array. Attributes which are language depended (like the title, keyword, description) will form an associated array with the key set to the language abbreviation. All other multiple attributes will form an indexed array. PHP functions never return object arrays.
- hw_document: This is a complete new data type which holds the actual document, e.g. HTML, PDF etc. It is somewhat optimised for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associated array with statistical information about them. The array is the last element of the object record array. The statistical array contains the following entries:

Hidden

Number of object records with attribute PresentationHints set to Hidden.

CollectionHead

Number of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHead

Number of object records with attribute PresentationHints set to FullCollectionHead.

CollectionHeadNr

Index in array of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHeadNr

Index in array of object records with attribute PresentationHints set to FullCollectionHead.

Total

Total: Number of object records.

Integration with Apache

The Hyperwave extension is best used when PHP is compiled as an Apache module. In such a case the underlying Hyperwave server can be hidden from users almost completely if Apache uses its rewriting engine. The following

instructions will explain this.

Since PHP with Hyperwave support built into Apache is intended to replace the native Hyperwave solution based on Wavemaster I will assume that the Apache server will only serve as a Hyperwave web interface. This is not necessary but it simplifies the configuration. The concept is quite simple. First of all you need a PHP script which evaluates the PATH_INFO variable and treats its value as the name of a Hyperwave object. Let's call this script 'Hyperwave'. The URL http://your.hostname/Hyperwave/name_of_object would than return the Hyperwave object with the name 'name_of_object'. Depending on the type of the object the script has to react accordingly. If it is a collection, it will probably return a list of children. If it is a document it will return the mime type and the content. A slight improvement can be achieved if the Apache rewriting engine is used. From the users point of view it would be more straight forward if the URL http://your.hostname/name_of_object would return the object. The rewriting rule is quite easy:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Now every URL relates to an object in the Hyperwave server. This causes a simple to solve problem. There is no way to execute a different script, e.g. for searching, than the 'Hyperwave' script. This can be fixed with another rewriting rule like the following:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

This will reserve the directory `/usr/local/apache/htdocs/hw` for additional scripts and other files. Just make sure this rule is evaluated before the one above. There is just a little drawback: all Hyperwave objects whose name starts with 'hw/' will be shadowed. So, make sure you don't use such names. If you need more directories, e.g. for images just add more rules or place them all in one directory. Finally, don't forget to turn on the rewriting engine with

```
RewriteEngine on
```

My experiences have shown that you will need the following scripts:

- to return the object itself
- to allow searching
- to identify yourself
- to set your profile
- one for each additional function like to show the object attributes, to show information about users, to show the status of the server, etc.

Todo

There are still some things todo:

- The hw_InsertDocument has to be split into **hw_InsertObject()** and **hw_PutDocument()**.
- The names of several functions are not fixed, yet.
- Most functions require the current connection as its first parameter. This leads to a lot of typing, which is quite often not necessary if there is just one open connection. A default connection will improve this.
- Conversion form object record into object array needs to handle any multiple attribute.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4)

convert attributes from object array to object record

```
strin hw_array2objrec (array object_array)
```

Converts an *object_array* into an object record. Multiple attributes like 'Title' in different languages are treated properly.

See also **hw_objrec2array()**.

hw_Children (PHP 3>= 3.0.3, PHP 4)

object ids of children

```
array hw_children (int connection, int objectID)
```

Returns an array of object ids. Each id belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4)

object records of children

```
array hw_childrenobj (int connection, int objectID)
```

Returns an array of object records. Each object record belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_Close (PHP 3>= 3.0.3, PHP 4)

closes the Hyperwave connection

```
int hw_close (int connection)
```

Returns false if connection is not a valid connection index, otherwise true. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect (PHP 3>= 3.0.3, PHP 4)

opens a connection

```
int hw_connect (string host, int port, string username, string password)
```

Opens a connection to a Hyperwave server and returns a connection index on success, or false if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also **hw_pConnect()**.

hw_Cp (PHP 3>= 3.0.3, PHP 4)

copies objects

```
int hw_cp (int connection, array object_id_array, int destination id)
```

Copies the objects with object ids as specified in the second parameter to the collection with the id *destination id*.

The value return is the number of copied objects.

See also **hw_mv()**.

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4)

deletes object

```
int hw_deleteobject (int connection, int object_to_delete)
```

Deletes the object with the given object id in the second parameter. It will delete all instances of the object.

Returns TRUE if no error occurs otherwise FALSE.

See also **hw_mv()**.

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4)

object id object belonging to anchor

```
int hw_docbyanchor (int connection, int anchorID)
```

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4)

object record object belonging to anchor

```
string hw_docbyanchorobj (int connection, int anchorID)
```

Returns an th object record of the document to which *anchorID* belongs.

hw_Document_Attributes (PHP 3>= 3.0.3, PHP 4)

object record of hw_document

```
string hw_document_attributes (int hw_document)
```

Returns the object record of the document.

For backward compatibility, **hw_DocumentAttributes()** is also accepted. This is deprecated, however.

See also **hw_Document_BodyTag()**, and **hw_Document_Size()**.

hw_Document_BodyTag (PHP 3>= 3.0.3, PHP 4)

body tag of hw_document

```
string hw_document_bodytag (int hw_document)
```

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also **hw_Document_Attributes()**, and **hw_Document_Size()**.

For backward compatibility, **hw_DocumentBodyTag()** is also accepted. This is deprecated, however.

hw_Document_Content (PHP 3>= 3.0.3, PHP 4)

returns content of hw_document

```
string hw_document_content (int hw_document)
```

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also **hw_Document_Attributes()**, **hw_Document_Size()**, and **hw_DocumentSetContent()**.

hw_Document_SetContent (PHP 4 >= 4.0b2)

sets/replaces content of hw_document

```
string hw_document_setcontent (int hw_document, string content)
```

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also **hw_Document_Attributes()**, **hw_Document_Size()**, and **hw_Document_Content()**.

hw_Document_Size (PHP 3>= 3.0.3, PHP 4)

size of hw_document

```
int hw_document_size (int hw_document)
```

Returns the size in bytes of the document.

See also **hw_Document_BodyTag()**, and **hw_Document_Attributes()**.

For backward compatibility, **hw_DocumentSize()** is also accepted. This is deprecated, however.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4)

returns error message

```
string hw_errormsg (int connection)
```

Returns a string containing the last error message or 'No Error'. If false is returned, this function failed. The message relates to the last command.

hw_EditText (PHP 3>= 3.0.3, PHP 4)

retrieve text document

```
int hw_edittext (int connection, int hw_document)
```

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_Document_BodyTag()**, **hw_Document_Size()**, **hw_Output_Document()**, **hw_GetText()**.

hw_Error (PHP 3>= 3.0.3, PHP 4)

error number

```
int hw_error (int connection)
```

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4)

frees hw_document

```
int hw_free_document (int hw_document)
```

Frees the memory occupied by the Hyperwave document.

hw_GetParents (PHP 3>= 3.0.3, PHP 4)

object ids of parents

```
array hw_getparents (int connection, int objectID)
```

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID *objectID*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4)

object records of parents

```
array hw_getparentsobj (int connection, int objectID)
```

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID *objectID*.

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4)

object ids of child collections

```
array hw_getchildcoll (int connection, int objectID)
```

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_GetChildren()**, and **hw_GetChildDocColl()**.

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4)

object records of child collections

```
array hw_getchildcollobj (int connection, int objectID)
```

Returns an array of object records. Each object records belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_ChildrenObj()**, and **hw_GetChildDocCollObj()**.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4)

Gets a remote document

```
int hw_getremote (int connection, int objectID)
```

Returns a remote document. Remote documents in Hyperwave notation are documents retrieved from an external source. Common remote documents are for example external web pages or queries in a database. In order to be able to access external sources throught remote documents Hyperwave introduces the HGI (Hyperwave Gateway Interface) which is similar to the CGI. Currently, only ftp, http-servers and some databases can be accessed by the HGI. Calling **hw_GetRemote()** returns the document from the external source. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also **hw_GetRemoteChildren()**.

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4)

Gets children of remote document

```
int hw_getremotechildren (int connection, string object record)
```

Returns the children of a remote document. Children of a remote document are remote documents itself. This makes sense if a database query has to be narrowed and is explained in Hyperwave Programmers' Guide. If the number of children is 1 the function will return the document itself formated by the Hyperwave Gateway Interface (HGI). If the number of children is greater than 1 it will return an array of object record with each maybe the input value for another call to **hw_GetRemoteChildren()**. Those object records are virtual and do not exist in the Hyperwave server, therefore

they do not have a valid object ID. How exactely such an object record looks like is up to the HGI. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also **hw_GetRemote()**.

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4)

Returns anchors pointing at object

```
array hw_getsrcbydestobj (int connection, int objectID)
```

Returns the object records of all anchors pointing to the object with ID *objectID*. The object can either be a document or an anchor of type destination.

See also **hw_GetAnchors()**.

hw_GetObject (PHP 3>= 3.0.3, PHP 4)

object record

```
array hw_getobject (int connection, [int|array] objectID, string query)
```

Returns the object record for the object with ID *objectID* if the second parameter is an integer. If the second parameter is an array of integer the function will return an array of object records. In such a case the last parameter is also evaluated which is a query string.

The query string has the following syntax:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NOT */

<expr> "||" <expr> | /* OR */

<expr> "&&" <expr> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* any attribute name (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* equal */

"<" | /* less than (string compare) */

">" | /* greater than (string compare) */

"~" /* regular expression matching */

The query allows to further select certain objects from the list of given objects. Unlike the other query functions, this query may use not indexed attributes. How many object records are returned depends on the query and if access to the object is allowed.

See also **hw_GetAndLock()**, and **hw_GetObjectByQuery()**.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4)

return bject record and lock object

```
string hw_getandlock (int connection, int objectID)
```

Returns the object record for the object with ID *objectID*. It will also lock the object, so other users cannot access it until it is unlocked.

See also **hw_Unlock()**, and **hw_GetObject()**.

hw_GetText (PHP 3>= 3.0.3, PHP 4)

retrieve text document

```
int hw_gettext (int connection, int objectID [, mixed rootID/prefix])
```

Returns the document with object ID *objectID*. If the document has anchors which can be inserted, they will be inserted already. The optional parameter *rootID/prefix* can be a string or an integer. If it is an integer it determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document. All links will have the prefix '/my_script.php3/'. If you do not want this you can set the optional parameter *rootID/prefix* to any prefix which is used instead. Is this case it has to be a string.

If *rootID/prefix* is an integer and unequal to 0 the link is constructed from all the names starting at the object with the id *rootID/prefix* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a-b-c-internet_movie' with '-' being the seperator between hierachy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_Document_BodyTag()**, **hw_Document_Size()**, and **hw_Output_Document()**.

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4)

search object

```
array hw_getobjectbyquery (int connection, string query, int max_hits)
```

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_GetObjectByQueryObj()**.

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4)

search object

```
array hw_getobjectbyqueryobj (int connection, string query, int max_hits)
```

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_GetObjectByQuery()**.

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4)

search object in collection

```
array hw_getobjectbyquerycoll (int connection, int objectID, string query, int max_hits)
```

Searches for objects in collection with ID *objectID* and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_GetObjectByQueryCollObj()**.

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4)

search object in collection

```
array hw_getobjectbyquerycollobj (int connection, int objectID, string query, int max_hits)
```

Searches for objects in collection with ID *objectID* and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_GetObjectByQueryColl()**.

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4)

object ids of child documents of collection

```
array hw_getchilddoccoll (int connection, int objectID)
```

Returns array of object ids for child documents of a collection.

See also **hw_GetChildren()**, and **hw_GetChildColl()**.

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4)

object records of child documents of collection

```
array hw_getchilddoccollobj (int connection, int objectID)
```

Returns an array of object records for child documents of a collection.

See also **hw_ChildrenObj()**, and **hw_GetChildCollObj()**.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4)

object ids of anchors of document

```
array hw_getanchors (int connection, int objectID)
```

Returns an array of object ids with anchors of the document with object ID *objectID*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4)

object records of anchors of document

```
array hw_getanchorsobj (int connection, int objectID)
```

Returns an array of object records with anchors of the document with object ID *objectID*.

hw_Mv (PHP 3>= 3.0.3, PHP 4)

moves objects

```
int hw_mv (int connection, array object id array, int source id, int destination id)
```

Moves the objects with object ids as specified in the second parameter from the collection with id *source id* to the collection with the id *destination id*. If the destination id is 0 the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted. If you want to delete all instances at once, use **hw_deleteobject()**.

The value return is the number of moved objects.

See also **hw_cp()**, and **hw_deleteobject()**.

hw_Identify (PHP 3>= 3.0.3, PHP 4)

identifies as user

```
int hw_identify (string username, string password)
```

Identifies as user with *username* and *password*. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also **hw_Connect()**.

hw_InCollections (PHP 3>= 3.0.3, PHP 4)

check if object ids in collections

```
array hw_incollections (int connection, array object_id_array, array
collection_id_array, int return_collections)
```

Checks whether a set of objects (documents or collections) specified by the *object_id_array* is part of the collections listed in *collection_id_array*. When the fourth parameter *return_collections* is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info (PHP 3>= 3.0.3, PHP 4)

info about connection

```
string hw_info (int connection)
```

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl (PHP 3>= 3.0.3, PHP 4)

insert collection

```
int hw_inscoll (int connection, int objectID, array object_array)
```

Inserts a new collection with attributes as in *object_array* into collection with object ID *objectID*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4)

insert document

```
int hw_insdoc (int connection, int parentID, string object_record, string text)
```

Inserts a new document with attributes as in *object_record* into collection with object ID *parentID*. This function inserts either an object record only or an object record and a pure ascii text in *text* if *text* is given. If you want to insert a general document of any kind use **hw_insertdocument()** instead.

See also **hw_InsertDocument()**, and **hw_InsColl()**.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4)

upload any document

```
int hw_insertdocument (int connection, int parent_id, int hw_document)
```

Uploads a document into the collection with *parent_id*. The document has to be created before with **hw_NewDocument()**. Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType. The functions returns the object id of the new document or false.

See also **hw_PipeDocument()**.

hw_InsertObject (PHP 3>= 3.0.3, PHP 4)

inserts an object record

```
int hw_insertobject (int connection, string object rec, string parameter)
```

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also **hw_PipeDocument()**, **hw_InsertDocument()**, **hw_InsDoc()**, and **hw_InsColl()**.

hw_mapid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Maps global id on virtual local id

```
int hw_mapid (int connection, int server id, int object id)
```

Maps a global object id on any hyperwave server, even those you did not connect to with **hw_connect()**, onto a virtual object id. This virtual object id can then be used as any other object id, e.g. to obtain the object record with **hw_getobject()**. The server id is the first part of the global object id (GOid) of the object which is actually the IP number as an integer.

Note: In order to use this function you will have to set the F_DISTRIBUTED flag, which can currently only be set at compile time in hg_comm.c. It is not set by default. Read the comment at the beginning of hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

modifies object record

```
int hw_modifyobject (int connection, int object_to_change, array remove, array add, int mode)
```

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID *object_to_change*. The first array *remove* is a list of attributes to remove. The second array *add* is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one. **hw_modifyobject()** will always remove the attributes before it adds attributes unless the value of the attribute to remove is not a string or array.

The last parameter determines if the modification is performed recursively. 1 means recurive modification. If some of the objects cannot be modified they will be skiped without notice. **hw_error()** may not indicate an error though some of the objects could not be modified.

The keys of both arrays are the attributes name. The value of each array element can either be an array, a string or anything else. If it is an array each attribute value is constructed by the key of each element plus a colon and the value of each element. If it is a string it is taken as the attribute value. An empty string will result in a complete removal of that attribute. If the value is neither a string nor an array but something else, e.g. an integer, no operation at all will be performed on the attribute. This is neccessary if you want to to add a completely new attribute not just a new value for an existing attribute. If the remove array contained an empty string for that attribute, the attribute would be tried to be removed which would fail since it doesn't exist. The following addition of a new value for that attribute would also fail. Setting the value for that attribute to e.g. 0 would not even try to remove it and the addition will work.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.

Příklad 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => "books");
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third parameter to an empty array. If the attribute is the first one with that name to add, set attribute value in the remove array to an integer.

Příklad 2. adding a completely new attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => 0);
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Poznámka: Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:

Příklad 3. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Příklad 4. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
$addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.

Příklad 5. removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Poznámka: This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.

Poznámka: If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

Poznámka: Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.

Poznámka: The 'Name' attribute is somewhat special. In some cases it cannot be complete removed. You will get an error message 'Change of base attribute' (not clear when this happens). Therefore you will always have to add a new Name first and than remove the old one.

Poznámka: You may not suround this function by calls to **hw_getandlock()** and **hw_unlock()**. **hw_modifyobject()** does this internally.

Returns TRUE if no error occurs otherwise FALSE.

hw_New_Document (PHP 3>= 3.0.3, PHP 4)

create new document

```
int hw_new_document (string object_record, string document_data, int document_size)
```

Returns a new Hyperwave document with document data set to *document_data* and object record set to *object_record*. The length of the *document_data* has to passed in *document_size*This function does not insert the document into the Hyperwave server.

See also **hw_FreeDocument()**, **hw_Document_Size()**, **hw_Document_BodyTag()**, **hw_Output_Document()**, and **hw_InsertDocument()**.

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4)

convert attributes from object record to object array

```
array hw_objrec2array (string object_record [, array format])
```

Converts an *object_record* into an object array. The keys of the resulting array are the attributes names. Multi-value attributes like 'Title' in different languages form its own array. The keys of this array are the left part to the colon of the attribute value. This left part must be two characters long. Other multi-value attributes without a prefix form an indexed array. If the optional parameter is missing the attributes 'Title', 'Description' and 'Keyword' are treated as language attributes and the attributes 'Group', 'Parent' and 'HtmlAttr' as non-prefixed multi-value attributes. By passing an array holding the type for each attribute you can alter this behaviour. The array is an associated array with the attribute name as its key and the value being one of HW_ATTR_LANG or HW_ATTR_NONE

See also **hw_array2objrec()**.

hw_Output_Document (PHP 3>= 3.0.3, PHP 4)

prints hw_document

```
int hw_output_document (int hw_document)
```

Prints the document without the BODY tag.

For backward compatibility, **hw_OutputDocument()** is also accepted. This is deprecated, however.

hw_pConnect (PHP 3>= 3.0.3, PHP 4)

make a persistent database connection

```
int hw_pconnect (string host, int port, string username, string password)
```

Returns a connection index on success, or false if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also **hw_Connect()**.

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4)

retrieve any document

```
int hw_pipedocument (int connection, int objectID)
```

Returns the Hyperwave document with object ID *objectID*. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also **hw_GetText()** for more on link insertion, **hw_FreeDocument()**, **hw_Document_Size()**, **hw_Document_BodyTag()**, and **hw_Output_Document()**.

hw_Root (PHP 3>= 3.0.3, PHP 4)

root object id

```
int hw_root ()
```

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_Unlock (PHP 3>= 3.0.3, PHP 4)

unlock object

```
int hw_unlock (int connection, int objectID)
```

Unlocks a document, so other users regain access.

See also **hw_GetAndLock()**.

hw_Who (PHP 3>= 3.0.3, PHP 4)

List of currently logged in users

```
int hw_who (int connection)
```

Returns an array of users currently logged into the Hyperwave server. Each entry in this array is an array itself containing the elements id, name, system, onSinceDate, onSinceTime, TotalTime and self. 'self' is 1 if this entry belongs to the user who initianted the request.

hw_getusername (PHP 3>= 3.0.3, PHP 4)

name of currently logged in user

```
string hw_getusername (int connection)
```

Returns the username of the connection.

XXXI. ICAP funkce

Pokud chcete používat tyto funkce, musíte PHP zkompilovat s `-with-icap`. To vyžaduje nainstalovanou icap knihovnu. Stáhněte si nejnovější verzi z http://icap.chek.com/, zkompilujte ji a nainstalujte.

icap_open (PHP 4 >= 4.0b4)

Otevřít ICAP spojení

```
stream icap_open (string calendar, string username, string password, string options)
```

Při úspěchu vrací ICAP stream, při chybě `false`.

icap_open() otevře ICAP spojení s daným *`calendar`* zdrojem dat. Pokud je přítomen argument *`options`*, předá tyto *`options`* danému mailboxu.

icap_close (unknown)

Zavřít ICAP stream

```
int icap_close (int icap_stream [, int flags])
```

Zavře daný ICAP stream.

icap_fetch_event (PHP 4 >= 4.0b4)

Získat událost z kalendářového stream

```
int icap_fetch_event (int stream_id, int event_id [, int options])
```

icap_fetch_event() získá událost z kalendářového streamu určenou argumtem *`event_id`*.

Vrací objekt události obsahující:

- int id - ID této události.
- int public - `true`, pokud je událost veřejná, `false`, pokud je soukromá
- string category - kategorie této události
- string title - titul této události
- string description - popis této události
- int alarm - kolik minut před touto událostí se má odeslat alarm/připomínka
- object start - Objekt obsahující datetime záznam.
- object end - Objekt obsahující datetime záznam.

Všechny datetime záznamy tvoří objekt, který obsahuje:

- int year - rok
- int month - měsíc
- int mday - den v měsíci
- int hour - hodinu
- int min - minuty
- int sec - sekundy

icap_list_events (PHP 4 >= 4.0RC1)

Vrátit seznam událostí mezi dvěma daty

```
array icap_list_events (int stream_id, int begin_date [, int end_date])
```

Vrací pole id událostí, které jsou mezi dvěma danými daty.

icap_list_events() přijímá počáteční a konečné datum kalendářového streamu. Vrací pole identifikátorů událostí, které jsou mezi těmito dvěma daty.

Všechny datetime záznamy tvoří objekt, který obsahuje:

- int year - rok
- int month - měsíc
- int mday - den v měsíci
- int hour - hodinu
- int min - minuty
- int sec - sekundy

icap_store_event (PHP 4 >= 4.0b4)

Uložit událost do ICAP kalendáře

```
string icap_store_event (int stream_id, object event)
```

icap_store_event() ukládá událost do ICAP kalendáře. Objekt události se skládá z:

- int public - `true`, pokud je událost veřejná, `false`, pokud je soukromá
- string category - kategorie této události
- string title - titul této události
- string description - popis této události
- int alarm - kolik minut před touto událostí se má odeslat alarm/připomínka
- object start - Objekt obsahující datetime záznam.
- object end - Objekt obsahující datetime záznam.

Všechny datetime záznamy tvoří objekt, který obsahuje:

- int year - rok
- int month - měsíc
- int mday - den v měsíci
- int hour - hodinu
- int min - minuty
- int sec - sekundy

Při úspěchu vrací `true`, při chybě `false`.

icap_delete_event (PHP 4 >= 4.0b4)

Delete an event from an ICAP calendar

```
string icap_delete_event (int sream_id, int uid)
```

icap_delete_event() maže událost *uid* z kalendáře.

Vrací true.

icap_snooze (PHP 4 >= 4.0b4)

Zapnout alarm

```
string icap_snooze (int stream_id, int uid)
```

icap_snooze() zapne alarm pro událost *uid*.

Vrací true.

icap_list_alarms (PHP 4 >= 4.0b4)

Vrátit seznam událostí, které mají v dané datum/čas puštěný alarm

```
int icap_list_alarms (int stream_id, array date, array time)
```

Vrací pole identifikátorů událostí, které mají v dané datum/čas puštěný alarm.

icap_list_alarms() function přijímá datum v kalendářovém streamu. Vrací pole identifikátorů událostí, které mají v dané datum/čas puštěný alarm.

Všechny datetime záznamy tvoří objekt, který obsahuje:

- int year - rok
- int month - měsíc
- int mday - den v měsíci
- int hour - hodinu
- int min - minuty
- int sec - sekundy

XXXII. Image functions

You can use the image functions in PHP to get the size of JPEG, GIF, PNG, and SWF images, and if you have the GD library (available at http://www.boutell.com/gd/) you will also be able to create and manipulate images.

The format of images you are able to manipulate depend on the version of gd you install, and any other libraries gd might need to access those image formats. Versions of gd older than gd-1.6 support gif format images, and do not support png, where versions greater than gd-1.6 support png, not gif.

In order to read and write images in jpeg format, you will need to obtain and install jpeg-6b (available at ftp://ftp.uu.net/graphics/jpeg/), and then recompile gd to make use of jpeg-6b. You will also have to compile PHP with `-with-jpeg-dir=/path/to/jpeg-6b`.

To add support for Type 1 fonts, you can install t1lib (available at ftp://ftp.neuroinformatik.ruhr-uni-bochum.de/pub/software/t1lib/), and then add `-with-t1lib[=dir]`.

GetImageSize (PHP 3, PHP 4)

Get the size of a GIF, JPEG, PNG or SWF image

```
array getimagesize (string filename [, array imageinfo])
```

The **GetImageSize()** function will determine the size of any GIF, JPG, PNG or SWF image file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML IMG tag.

Returns an array with 4 elements. Index 0 contains the width of the image in pixels. Index 1 contains the height. Index 2 a flag indicating the type of the image. 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF. Index 3 is a text string with the correct "height=xxx width=xxx" string that can be used directly in an IMG tag.

Příklad 1. GetImageSize (file)

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Příklad 2. GetImageSize (URL)

```
<?php $size = GetImageSize ("http://www.php.net/gifs/logo.gif"); ?>
```

The optional *imageinfo* parameter allows you to extract some extended information from the image file. Currently this will return the diffrent JPG APP markers in an associative Array. Some Programs use these APP markers to embedd text information in images. A very common one in to embed IPTC http://www.iptc.org/ information in the APP13 marker. You can use the **iptcparse()** function to parse the binary APP13 marker into something readable.

Příklad 3. GetImageSize returning IPTC

```
<?php
    $size = GetImageSize ("testimg.jpg",&$info);
    if (isset ($info["APP13"])) {
        $iptc = iptcparse ($info["APP13"]);
        var_dump ($iptc);
    }
?>
```

Poznámka: This function does not require the GD image library.

Poznámka: URL support was added in PHP 4.0.5

ImageAlphaBlending (unknown)

Set the blending mode for an image

```
int imagealphablending (resource im, bool blendmode)
```

ImageAlphaBlending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as **ImageSetPixel()** determines how much of

the underlying color shouldbe allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *blendmode* is true, then blending mode is enabled, otherwise disabled.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageArc (PHP 3, PHP 4)

Draw a partial ellipse

```
int imagearc (int im, int cx, int cy, int w, int h, int s, int e, int col)
```

ImageArc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by im. *w* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e*. arguments.

ImageArc (PHP 3, PHP 4)

Draw a partial ellipse and fill it

```
int imagefilledarc (int im, int cx, int cy, int w, int h, int s, int e, int col, int style)
```

ImageFilledArc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by im. *w* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e*. arguments. *style* is a bitwise OR of the following possibilities:

1. IMG_ARC_PIE
2. IMG_ARC_CHORD
3. IMG_ARC_NOFILL
4. IMG_ARC_EDGED

IMG_ARC_PIE and IMG_ARC_CHORD are mutually exclusive; IMG_ARC_CHORD just connects the starting and ending angles with a straight line, while IMG_ARC_PIE produces a rounded edge. IMG_ARC_NOFILL indicates that the arc or chord should be outlined, not filled. IMG_ARC_EDGED, used together with IMG_ARC_NOFILL, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageEllipse (unknown)

Draw an ellipse

```
int imageellipse (resource im, int cx, int cy, int w, int h, int col)
```

ImageEllipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *im*. *w* and *h* specifies the ellipse's width and height respectively. The color of the ellipse is specified by *color*.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.2 or later

ImageFilledEllipse (unknown)

Draw a filled ellipse

```
int imagefilledellipse (resource im, int cx, int cy, int w, int h, int col)
```

ImageFilledEllipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *im*. *W* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageChar (PHP 3, PHP 4)

Draw a character horizontally

```
int imagechar (int im, int font, int x, int y, string c, int col)
```

ImageChar() draws the first character of *c* in the image identified by *id* with its upper-left at *x*,*y* (top left is 0, 0) with the color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used (with higher numbers corresponding to larger fonts).

See also **imageloadfont()**.

ImageCharUp (PHP 3, PHP 4)

Draw a character vertically

```
int imagecharup (int im, int font, int x, int y, string c, int col)
```

ImageCharUp() draws the character *c* vertically in the image identified by *im* at coordinates *x*, *y* (top left is 0, 0) with the color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

ImageColorAllocate (PHP 3, PHP 4)

Allocate a color for an image

```
int imagecolorallocate (int im, int red, int green, int blue)
```

ImageColorAllocate() returns a color identifier representing the color composed of the given RGB components. The *im* argument is the return from the **imagecreate()** function. **ImageColorAllocate()** must be called to create each color that is to be used in the image represented by *im*.

```
$white = ImageColorAllocate ($im, 255, 255, 255);
$black = ImageColorAllocate ($im, 0, 0, 0);
```

ImageColorDeAllocate (PHP 3>= 3.0.6, PHP 4)

De-allocate a color for an image

```
int imagecolordeallocate  (int im, int index)
```

The **ImageColorDeAllocate()** function de-allocates a color previously allocated with the **ImageColorAllocate()** function.

```
$white = ImageColorAllocate($im, 255, 255, 255);
ImageColorDeAllocate($im, $white);
```

ImageColorAt (PHP 3, PHP 4)

Get the index of the color of a pixel

```
int imagecolorat (int im, int x, int y)
```

Returns the index of the color of the pixel at the specified location in the image.

See also **imagecolorset()** and **imagecolorsforindex()**.

ImageColorClosest (PHP 3, PHP 4)

Get the index of the closest color to the specified color

```
int imagecolorclosest (int im, int red, int green, int blue)
```

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value.

The "distance" between the desired color and each color in the palette is calculated as if the RGB values represented points in three-dimensional space.

See also **imagecolorexact()**.

ImageColorClosestAlpha (unknown)

Get the index of the closest color to the specified color + alpha

```
int imagecolorclosestalpha (resource im, int red, int green, int blue, int alpha)
```

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and *alpha* level.

See also **imagecolorexactalpha()**.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageColorExact (PHP 3, PHP 4)

Get the index of the specified color

```
int imagecolorexact (int im, int red, int green, int blue)
```

Returns the index of the specified color in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also **imagecolorclosest()**.

ImageColorExactAlpha (unknown)

Get the index of the specified color + alpha

```
int imagecolorexactalpha (resource im, int red, int green, int blue, int alpha)
```

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also **imagecolorclosestalpha()**.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageColorResolve (PHP 3>= 3.0.2, PHP 4)

Get the index of the specified color or its closest possible alternative

```
int imagecolorresolve (int im, int red, int green, int blue)
```

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also **imagecolorclosest()**.

ImageColorResolveAlpha (unknown)

Get the index of the specified color + alpha or its closest possible alternative

```
int imagecolorresolvealpha (resource im, int red, int green, int blue, int alpha)
```

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also **imagecolorclosestalpha()**.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageGammaCorrect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Apply a gamma correction to a GD image

```
int imagegammacorrect  (int im, double inputgamma, double outputgamma)
```

The **ImageGammaCorrect()** function applies gamma correction to a gd image stream (*im*) given an input gamma, the parameter *inputgamma* and an output gamma, the parameter *outputgamma*.

ImageColorSet (PHP 3, PHP 4)

Set the color for the specified palette index

```
bool imagecolorset (int im, int index, int red, int green, int blue)
```

This sets the specified index in the palette to the specified color. This is useful for creating flood-fill-like effects in paletted images without the overhead of performing the actual flood-fill.

See also **imagecolorat()**.

ImageColorsForIndex (PHP 3, PHP 4)

Get the colors for an index

```
array imagecolorsforindex (int im, int index)
```

This returns an associative array with red, green, and blue keys that contain the appropriate values for the specified color index.

See also **imagecolorat()** and **imagecolorexact()**.

ImageColorsTotal (PHP 3, PHP 4)

Find out the number of colors in an image's palette

```
int imagecolorstotal (int im)
```

This returns the number of colors in the specified image's palette.

See also **imagecolorat()** and **imagecolorsforindex()**.

ImageColorTransparent (PHP 3, PHP 4)

Define a color as transparent

```
int imagecolortransparent (int im [, int col])
```

ImageColorTransparent() sets the transparent color in the *im* image to *col*. *Im* is the image identifier returned by **ImageCreate()** and *col* is a color identifier returned by **ImageColorAllocate()**.

The identifier of the new (or current, if none is specified) transparent color is returned.

ImageCopy (PHP 3>= 3.0.6, PHP 4)

Copy part of an image

```
int ImageCopy  (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int src_y, int
src_w, int src_h)
```

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*.

ImageCopyMerge (PHP 4 >= 4.0.1)

Copy and merge part of an image

```
int ImageCopyMerge  (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int
src_y, int src_w, int src_h, int pct)
```

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **ImageCopy()**.

ImageCopyMergeGray (unknown)

Copy and merge part of an image with gray scale

```
int ImageCopyMergeGray  (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int
src_y, int src_w, int src_h, int pct)
```

This function is identical to **ImageCopyMerge()** except that when merging it preservese the hue of the source by converting the destination pixels to gray scale before the copy operation.

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **ImageCopy()**.

This function is identical to **ImageCopyMerge()** except that when merging it preservese the hue of the source by converting the destination pixels to gray scale before the copy operation.

Poznámka: This function was added in PHP 4.0.6

ImageCopyResized (PHP 3, PHP 4)

Copy and resize part of an image

```
int imagecopyresized (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY,
int dstW, int dstH, int srcW, int srcH)
```

ImageCopyResized() copies a rectangular portion of one image to another image. *Dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left

corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

See also **ImageCopyResampled()**.

ImageCopyResampled (unknown)

Copy and resize part of an image with resampling

```
int imagecopyresampled (resource dst_im, resource src_im, int dstX, int dstY, int srcX,
int srcY, int dstW, int dstH, int srcW, int srcH)
```

ImageCopyResampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity. *Dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

See also **ImageCopyResized()**.

Poznámka: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageCreate (PHP 3, PHP 4)

Create a new palette based image

```
int imagecreate (int x_size, int y_size)
```

ImageCreate() returns an image identifier representing a blank image of size *x_size* by *y_size*.

Příklad 1. Creating a new GD image stream and outputting an image.

```
<?php
header ("Content-type: image/png");
$im = @ImageCreate (50, 100)
    or die ("Cannot Initialize new GD image stream");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5,  "A Simple Text String", $text_color);
ImagePng ($im);
?>
```

ImageCreateTrueColor (unknown)

Create a new true color image

```
resource imagecreatetruecolor (int x_size, int y_size)
```

ImageCreateTrueColor() returns an image identifier representing a black image of size *x_size* by *y_size*.

Poznámka: This function was added in PHP 4.0.6

Poznámka: This function requires GD 2.0.1 or later

ImageTrueColorToPalette (unknown)

Convert a true color image to a palette image

```
void imagetruecolortopalette (resource im, bool dither, int ncolors)
```

ImageTrueColorToPalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

dither indicates if the image should be dithered - if it is true then dithering will be used which will result in a more speckled image but with better color approximation.

ncolors sets the maximum number of colors that should be retained in the palette.

Poznámka: This function was added in PHP 4.0.6

Poznámka: This function requires GD 2.0.1 or later

ImageCreateFromGIF (PHP 3, PHP 4)

Create a new image from file or URL

```
int imagecreatefromgif (string filename)
```

ImageCreateFromGif() returns an image identifier representing the image obtained from the given filename.

ImageCreateFromGif() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error GIF:

Příklad 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadGif ($imgname) {
    $im = @ImageCreateFromGIF ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        ImageString($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Poznámka: Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.

ImageCreateFromJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0RC1)

Create a new image from file or URL

```
int imagecreatefromjpeg (string filename)
```

ImageCreateFromJPEG() returns an image identifier representing the image obtained from the given filename.

ImagecreateFromJPEG() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error JPEG:

Příklad 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadJpeg ($imgname) {
    $im = @ImageCreateFromJPEG ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        ImageString ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromPNG (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Create a new image from file or URL

```
int imagecreatefrompng (string filename)
```

ImageCreateFromPNG() returns an image identifier representing the image obtained from the given filename.

ImageCreateFromPNG() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error PNG:

Příklad 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadPNG ($imgname) {
    $im = @ImageCreateFromPNG ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = ImageCreate (150, 30); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        ImageString ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromWBMP (PHP 4 >= 4.0.1)

Create a new image from file or URL

```
int imagecreatefromwbmp (string filename)
```

ImageCreateFromWBMP() returns an image identifier representing the image obtained from the given filename.

ImageCreateFromWBMP() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Příklad 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadWBMP ($imgname) {
    $im = @ImageCreateFromWBMP ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = ImageCreate (20, 20); /* Create a blank image */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        ImageString ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromString (PHP 4 >= 4.0.4)

Create a new image from the image stream in the string

```
int imagecreatefromstring (string image)
```

ImageCreateFromString() returns an image identifier representing the image obtained from the given string.

ImageDashedLine (PHP 3, PHP 4)

Draw a dashed line

```
int imagedashedline (int im, int x1, int y1, int x2, int y2, int col)
```

ImageDashedLine() draws a dashed line from *x1*, *y1* to *x2*, *y2* (top left is 0, 0) in image *im* of color *col*.

See also **ImageLine()**.

ImageDestroy (PHP 3, PHP 4)

Destroy an image

```
int imagedestroy (int im)
```

ImageDestroy() frees any memory associated with image *im*. *Im* is the image identifier returned by the **ImageCreate()** function.

ImageFill (PHP 3, PHP 4)

Flood fill

```
int imagefill (int im, int x, int y, int col)
```

ImageFill() performs a flood fill starting at coordinate *x*, *y* (top left is 0, 0) with color *col* in the image *im*.

ImageFilledPolygon (PHP 3, PHP 4)

Draw a filled polygon

```
int imagefilledpolygon (int im, array points, int num_points, int col)
```

ImageFilledPolygon() creates a filled polygon in image *im*. *Points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *Num_points* is the total number of vertices.

ImageFilledRectangle (PHP 3, PHP 4)

Draw a filled rectangle

```
int imagefilledrectangle (int im, int x1, int y1, int x2, int y2, int col)
```

ImageFilledRectangle() creates a filled rectangle of color **col()** in image *im* starting at upper left coordinates *x1*, *y1* and ending at bottom right coordinates *x2*, *y2*. 0, 0 is the top left corner of the image.

ImageFillToBorder (PHP 3, PHP 4)

Flood fill to specific color

```
int imagefilltoborder (int im, int x, int y, int border, int col)
```

ImageFillToBorder() performs a flood fill whose border color is defined by *border*. The starting point for the fill is *x*, *y* (top left is 0, 0) and the region is filled with color *col*.

ImageFontHeight (PHP 3, PHP 4)

Get font height

```
int imagefontheight (int font)
```

Returns the pixel height of a character in the specified font.

See also **ImageFontWidth()** and **ImageLoadFont()**.

ImageFontWidth (PHP 3, PHP 4)

Get font width

```
int imagefontwidth (int font)
```

Returns the pixel width of a character in font.

See also **ImageFontHeight()** and **ImageLoadFont()**.

ImageGIF (PHP 3, PHP 4)

Output image to browser or file

```
int imagegif (int im [, string filename])
```

ImageGIF() creates the GIF file in filename from the image *im*. The *im* argument is the return from the **imagecreate()** function.

The image format will be GIF87a unless the image has been made transparent with **ImageColorTransparent()**, in which case the image format will be GIF89a.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/gif content-type using **header()**, you can create a PHP script that outputs GIF images directly.

Poznámka: Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.

The following code snippet allows you to write more portable PHP applications by auto-detecting the type of GD support which is available. Replace the sequence `Header("Content-type: image/gif"); ImageGIF($im);` by the more flexible sequence:

```
<?php
  if (function_exists("imagegif")) {
    Header("Content-type: image/gif");
    ImageGIF($im);
  }
  elseif (function_exists("imagejpeg")) {
    Header("Content-type: image/jpeg");
    ImageJPEG($im, "", 0.5);
  }
  elseif (function_exists("imagepng")) {
    Header("Content-type: image/png");
    ImagePNG($im);
  }
  elseif (function_exists("imagewbmp")) {
    Header("Content-type: image/vnd.wap.wbmp");
    ImageWBMP($im);
  }
  else
    die("No image support in this PHP server");
?>
```

Poznámka: As of version 3.0.18 and 4.0.2 you can use the function **imagetypes()** in place of **function_exists()** for checking the presence of the various supported image formats:

```
if (ImageTypes() & IMG_GIF) {
    Header("Content-type: image/gif");
    ImageGif($im);
}
elseif (ImageTypes() & IMG_JPG) {
... etc.
```

See also **ImagePNG()**, **ImageWBMP()**, **ImageJPEG()**, **ImageTypes()**.

ImagePNG (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Output a PNG image to either the browser or a file

```
int imagepng  (int im [, string filename])
```

The **ImagePNG()** outputs a GD image stream (*im*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.

```
<?php
$im = ImageCreateFromPng("test.png");
ImagePng($im);
?>
```

See also **ImageGIF()**, **ImageWBMP()**, **ImageJPEG()**, **ImageTypes()**.

ImageJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0RC1)

Output image to browser or file

```
int imagejpeg (int im [, string filename [, int quality]])
```

ImageJPEG() creates the JPEG file in filename from the image *im*. The *im* argument is the return from the **ImageCreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. To skip the filename argument in order to provide a quality argument just use an empty string (''). By sending an image/jpeg content-type using **header()**, you can create a PHP script that outputs JPEG images directly.

Poznámka: JPEG support is only available in PHP if PHP was compiled against GD-1.8 or later.

See also **ImagePNG()**, **ImageGIF()**, **ImageWBMP()**, **ImageTypes()**.

ImageWBMP (PHP 3>= 3.0.15, PHP 4 >= 4.0.1)

Output image to browser or file

```
int imageWBMP (int im [, string filename])
```

ImageWBMP() creates the WBMP file in filename from the image *im*. The *im* argument is the return from the **ImageCreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using **header()**, you can create a PHP script that outputs WBMP images directly.

Poznámka: WBMP support is only available in PHP if PHP was compiled against GD-1.8 or later.

See also **ImagePNG()**, **ImageGIF()**, **ImageJPEG()**, **ImageTypes()**.

ImageInterlace (PHP 3, PHP 4)

Enable or disable interlace

```
int imageinterlace (int im [, int interlace])
```

ImageInterlace() turns the interlace bit on or off. If interlace is 1 the im image will be interlaced, and if interlace is 0 the interlace bit is turned off.

This functions returns whether the interlace bit is set for the image.

ImageLine (PHP 3, PHP 4)

Draw a line

```
int imageline (int im, int x1, int y1, int x2, int y2, int col)
```

ImageLine() draws a line from *x1*, *y1* to *x2*, *y2* (top left is 0, 0) in image im of color *col*.

See also **ImageCreate()** and **ImageColorAllocate()**.

ImageLoadFont (PHP 3, PHP 4)

Load a new font

```
int imageloadfont (string file)
```

ImageLoadFont() loads a user-defined bitmap font and returns an identifier for the font (that is always greater than 5, so it will not conflict with the built-in fonts).

The font file format is currently binary and architecture dependent. This means you should generate the font files on the same type of CPU as the machine you are running PHP on.

Tabulka 1. Font file format

byte position	C data type	description
byte 0-3	int	number of characters in the font
byte 4-7	int	value of first character in the font (often 32 for space)
byte 8-11	int	pixel width of each character
byte 12-15	int	pixel height of each character
byte 16-	char	array with character data, one byte per pixel in each character, for a total of (nchars*width*height) bytes.

See also **ImageFontWidth()** and **ImageFontHeight()**.

ImagePolygon (PHP 3, PHP 4)

Draw a polygon

```
int imagepolygon (int im, array points, int num_points, int col)
```

ImagePolygon() creates a polygon in image id. *Points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *Num_points* is the total number of vertices.

See also **imagecreate()**.

ImagePSBBox (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Give the bounding box of a text rectangle using PostScript Type1 fonts

```
array imagepsbbox (string text, int font, int size [, int space [, int tightness [,
float angle]]])
```

Size is expressed in pixels.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, and *angle* are optional.

The bounding box is calculated using information available from character metrics, and unfortunately tends to differ slightly from the results achieved by actually rasterizing the text. If the angle is 0 degrees, you can expect the text to need 1 pixel more to every direction.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also **imagepstext()**.

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Change the character encoding vector of a font

```
int imagepsencodefont (int font_index, string encodingfile)
```

Loads a character encoding vector from from a file and changes the fonts encoding vector to it. As a PostScript fonts default vector lacks most of the character positions above 127, you'll definitely want to change this if you use an other language than english. The exact format of this file is described in T1libs documentation. T1lib comes with two ready-to-use files, IsoLatin1.enc and IsoLatin2.enc.

If you find yourself using this function all the time, a much better way to define the encoding is to set ps.default_encoding in the configuration file to point to the right encoding file and all fonts you load will automatically have the right encoding.

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Free memory used by a PostScript Type 1 font

```
void imagepsfreefont (int fontindex)
```

See also **ImagePSLoadFont()**.

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Load a PostScript Type 1 font from file

```
int imagepsloadfont (string filename)
```

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns false and prints a message describing what went wrong.

See also **ImagePSFreeFont()**.

ImagePsExtendFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Extend or condense a font

```
bool imagepsextendfont  (int font_index, double extend)
```

Extend or condense a font (*font_index*), if the value of the *extend* parameter is less than one you will be condensing the font.

ImagePsSlantFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Slant a font

```
bool imagepsslantfont  (int font_index, double slant)
```

Slant a font given by the *font_index* parameter with a slant of the value of the *slant* parameter.

ImagePSText (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

To draw a text string over an image using PostScript Type1 fonts

```
array imagepstext (int image, string text, int font, int size, int foreground, int
background, int x, int y [, int space [, int tightness [, float angle [, int
antialias_steps]]]])
```

Size is expressed in pixels.

Foreground is the color in which the text will be painted. *Background* is the color to which the text will try to fade in with antialiasing. No pixels with the color *background* are actually painted, so the background image does not need to be of solid color.

The coordinates given by *x*, *y* will define the origin (or reference point) of the first character (roughly the lower-left corner of the character). This is different from the **ImageString()**, where *x*, *y* define the upper-right corner of the first character. Refer to PostScipt documentation about fonts and their measuring system if you have trouble understanding how this works.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Antialias_steps allows you to control the number of colours used for antialiasing text. Allowed values are 4 and 16. The higher value is recommended for text sizes lower than 20, where the effect in text quality is quite visible. With bigger sizes, use 4. It's less computationally intensive.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, *angle* and *antialias* are optional.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also **imagepsbbox()**.

ImageRectangle (PHP 3, PHP 4)

Draw a rectangle

```
int imagerectangle (int im, int x1, int y1, int x2, int y2, int col)
```

ImageRectangle() creates a rectangle of color col in image im starting at upper left coordinate x1, y1 and ending at bottom right coordinate x2, y2. 0, 0 is the top left corner of the image.

ImageSetPixel (PHP 3, PHP 4)

Set a single pixel

```
int imagesetpixel (int im, int x, int y, int col)
```

ImageSetPixel() draws a pixel at *x*, *y* (top left is 0, 0) in image *im* of color *col*.

See also **ImageCreate()** and **ImageColorAllocate()**.

ImageSetBrush (unknown)

Set the brush image for line drawing

```
int imagesetbrush (resource im, resource brush)
```

ImageSetBrush() sets the brush image to be used by all line drawing functions (such as **ImageLine()** and **ImagePolygon()**) when drawing with the special colors IMG_COLOR_BRUSHED or IMG_COLOR_STYLEDBRUSHED.

> **Poznámka:** You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the IMG_COLOR_BRUSHED or IMG_COLOR_STYLEDBRUSHED colors until you have set a new brush image!

> **Poznámka:** This function was added in PHP 4.0.6

ImageSetTile (unknown)

Set the tile image for filling

```
int imagesettile (resource im, resource tile)
```

ImageSetTile() sets the tile image to be used by all region filling functions (such as **ImageFill()** and **ImageFilledPolygon()**) when filling with the special color IMG_COLOR_TILED.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with **ImageColorTransparent()**, a tile allows certain parts of the underlying area to shine through can be created.

> **Poznámka:** You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the IMG_COLOR_TILED color until you have set a new tile image!

> **Poznámka:** This function was added in PHP 4.0.6

ImageSetThickness (unknown)

Set the thickness for line drawing

```
void imagesetthickness (resource im, int thickness)
```

ImageSetThickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to `thickness` pixels.

> **Poznámka:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageString (PHP 3, PHP 4)

Draw a string horizontally

```
int imagestring (int im, int font, int x, int y, string s, int col)
```

ImageString() draws the string *s* in the image identified by *im* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **ImageLoadFont()**.

ImageStringUp (PHP 3, PHP 4)

Draw a string vertically

```
int imagestringup (int im, int font, int x, int y, string s, int col)
```

ImageStringUp() draws the string *s* vertically in the image identified by *im* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **ImageLoadFont()**.

ImageSX (PHP 3, PHP 4)

Get image width

```
int imagesx (int im)
```

ImageSX() returns the width of the image identified by *im*.

See also **ImageCreate()** and **ImageSY()**.

ImageSY (PHP 3, PHP 4)

Get image height

```
int imagesy (int im)
```

ImageSY() returns the height of the image identified by *im*.

See also **ImageCreate()** and **ImageSX()**.

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4)

Give the bounding box of a text using TypeType fonts

```
array imagettfbbox (int size, int angle, string fontfile, string text)
```

This function calculates and returns the bounding box in pixels for a TrueType text.

text

The string to be measured.

size

The font size in pixels.

fontfile

The name of the TrueType font file. (Can also be an URL.)

angle

Angle in degrees in which *text* will be measured.

ImageTTFBBox() returns an array with 8 elements representing four points making the bounding box of the text:

0	lower left corner, X position
1	lower left corner, Y position
2	lower right corner, X position
3	lower right corner, Y position
4	upper right corner, X position
5	upper right corner, Y position
6	upper left corner, X position
7	upper left corner, Y position

The points are relative to the *text* regardless of the angle, so "upper left" means in the top left-hand corner seeing the text horizontallty.

This function requires both the GD library and the FreeType library.

See also **ImageTTFText()**.

ImageTTFText (PHP 3, PHP 4)

Write text to the image using TrueType fonts

```
array imagettftext (int im, int size, int angle, int x, int y, int col, string
fontfile, string text)
```

ImageTTFText() draws the string *text* in the image identified by *im*, starting at coordinates *x*, *y* (top left is 0, 0), at an angle of *angle* in color *col*, using the TrueType font file identified by *fontfile*.

The coordinates given by *x*, *y* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the **ImageString()**, where x, y define the upper-right corner of the first character.

Angle is in degrees, with 0 degrees being left-to-right reading text (3 o'clock direction), and higher values representing a counter-clockwise rotation. (i.e., a value of 90 would result in bottom-to-top reading text).

Fontfile is the path to the TrueType font you wish to use.

Text is the text string which may include UTF-8 character sequences (of the form: {) to access characters in a font beyond the first 255.

Col is the color index. Using the negative of a color index has the effect of turning off antialiasing.

ImageTTFText() returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is upper left, upper right, lower right, lower left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontallty.

This example script will produce a black GIF 400x30 pixels, with the words "Testing..." in white in the font Arial.

Příklad 1. ImageTTFText

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
              "Testing... Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

This function requires both the GD library and the FreeType (http://www.freetype.org/) library.

See also **ImageTTFBBox()**.

ImageTypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Return the image types supported by this PHP build

```
int imagetypes(void);
```

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, IMG_GIF | IMG_JPG | IMG_PNG | IMG_WBMP. To check for PNG support, for example, do this:

Příklad 1. ImageTypes

```
<?php
if (ImageTypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

read_exif_data (PHP 4 >= 4.0.1)

Read the EXIF headers from a JPEG

```
array read_exif_data (string filename)
```

The **read_exif_data()** function reads the EXIF headers from a JPEG image file. It returns an associative array where the indexes are the Exif header names and the values are the values associated with those Exif headers. Exif headers tend to be present in JPEG images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can’t always rely on a specific Exif header being present.

Příklad 1. read_exif_data

```
<?php
 $exif = read_exif_data (’p0001807.jpg’);
 while(list($k,$v)=each($exif)) {
   echo "$k: $v<br>\n";
 }
?>

Output:
FileName: p0001807.jpg
FileDateTime: 929353056
FileSize: 378599
CameraMake: Eastman Kodak Company
CameraModel: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
DateTime: 1999:06:14 01:37:36
Height: 1024
Width: 1536
IsColor: 1
FlashUsed: 0
FocalLength:  8.0mm
RawFocalLength: 8
ExposureTime:  0.004 s (1/250)
```

```
RawExposureTime: 0.0040000001899898
ApertureFNumber: f/ 9.5
RawApertureFNumber: 9.5100002288818
FocusDistance: 16.66m
RawFocusDistance: 16.659999847412
Orientation: 1
ExifVersion: 0200
```

Poznámka: This function is only available in PHP 4 compiled using –enable-exif

This function does not require the GD image library.

XXXIII. IMAP, POP3 and NNTP functions

To get these functions to work, you have to compile PHP with `-with-imap`. That requires the c-client library to be installed. Grab the latest version from ftp://ftp.cac.washington.edu/imap/ and compile it. Then copy `c-client/c-client.a` to `/usr/local/lib/libc-client.a` or some other directory on your link path and copy `c-client/rfc822.h`, `mail.h` and `linkage.h` to `/usr/local/include` or some other directory in your include path.

Note that these functions are not limited to the IMAP protocol, despite their name. The underlying c-client library also supports NNTP, POP3 and local mailbox access methods.

This document can't go into detail on all the topics touched by the provided functions. Further information is provided by the documentation of the c-client library source (`docs/internal.txt`). and the following RFC documents:

- RFC821 (http://www.faqs.org/rfcs/rfc821.html): Simple Mail Transfer Protocol (SMTP).
- RFC822 (http://www.faqs.org/rfcs/rfc822.html): Standard for ARPA internet text messages.
- RFC2060 (http://www.faqs.org/rfcs/rfc2060.html): Internet Message Access Protocol (IMAP) Version 4rev1.
- RFC1939 (http://www.faqs.org/rfcs/rfc1939.html): Post Office Protocol Version 3 (POP3).
- RFC977 (http://www.faqs.org/rfcs/rfc977.html): Network News Transfer Protocol (NNTP).
- RFC2076 (http://www.faqs.org/rfcs/rfc2076.html): Common Internet Message Headers.
- RFC2045 (http://www.faqs.org/rfcs/rfc2045.html) , RFC2046 (http://www.faqs.org/rfcs/rfc2046.html) , RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) , RFC2048 (http://www.faqs.org/rfcs/rfc2048.html) & RFC2049 (http://www.faqs.org/rfcs/rfc2049.html): Multipurpose Internet Mail Extensions (MIME).

A detailed overview is also available in the books Programming Internet Email (http://www.oreilly.com/catalog/progintemail/noframes.html) by David Wood and Managing IMAP (http://www.oreilly.com/catalog/mimap/noframes.html) by Dianna Mullet & Kevin Mullet.

imap_8bit (PHP 3, PHP 4)

Convert an 8bit string to a quoted-printable string

```
string imap_8bit (string string)
```

Convert an 8bit string to a quoted-printable string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns a quoted-printable string.

See also **imap_qprint()**.

imap_alerts (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

This function returns all IMAP alert messages (if any) that have occurred during this page request or since the alert stack was reset

```
array imap_alerts (void)
```

This function returns an array of all of the IMAP alert messages generated since the last **imap_alerts()** call, or the beginning of the page. When **imap_alerts()** is called, the alert stack is subsequently cleared. The IMAP specification requires that these messages be passed to the user.

imap_append (PHP 3, PHP 4)

Append a string message to a specified mailbox

```
int imap_append (int imap_stream, string mbox, string message [, string flags])
```

Returns true on sucess, false on error.

imap_append() appends a string message to the specified mailbox *mbox*. If the optional *flags* is specified, writes the *flags* to that mailbox also.

When talking to the Cyrus IMAP server, you must use "\r\n" as your end-of-line terminator instead of "\n" or the operation will fail.

Příklad 1. imap_append() example

```
$stream = imap_open("{your.imap.host}INBOX.Drafts","username", "password");

$check = imap_check($stream);
print "Msg Count before append: ". $check->Nmsgs."\n";

imap_append($stream,"{your.imap.host}INBOX.Drafts"
                  ,"From: me@my.host\r\n"
                  ."To: you@your.host\r\n"
                  ."Subject: test\r\n"
                  ."\r\n"
                  ."this is a test message, please ignore\r\n"
                  );

$check = imap_check($stream);
print "Msg Count after append : ". $check->Nmsgs."\n";

imap_close($stream);
```

imap_base64 (PHP 3, PHP 4)

Decode BASE64 encoded text

```
string imap_base64 (string text)
```

imap_base64() function decodes BASE-64 encoded text (see RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8). The decoded message is returned as a string.

See also **imap_binary()**.

imap_binary (PHP 3>= 3.0.2, PHP 4)

Convert an 8bit string to a base64 string

```
string imap_binary (string string)
```

Convert an 8bit string to a base64 string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8).

Returns a base64 string.

See also **imap_base64()**.

imap_body (PHP 3, PHP 4)

Read the message body

```
string imap_body (int imap_stream, int msg_number [, int flags])
```

imap_body() returns the body of the message, numbered *msg_number* in the current mailbox. The optional *flags* are a bit mask with one or more of the following:

- FT_UID - The *msgno* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in internal format, will not canonicalize to CRLF.

imap_body() will only return a verbatim copy of the message body. To extract single parts of a multipart MIME-encoded message you have to use **imap_fetchstructure()** to analyze its structure and **imap_fetchbody()** to extract a copy of a single body component.

imap_check (PHP 3, PHP 4)

Check current mailbox

```
object imap_check (int imap_stream)
```

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_check()** function checks the current mailbox status on the server and returns the information in an object with following properties:

- Date - last change of mailbox contents
- Driver - protocol used to access this mailbox: POP3, IMAP, NNTP
- Mailbox - the mailbox name
- Nmsgs - number of messages in the mailbox
- Recent - number of recent messages in the mailbox

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4)

Clears flags on messages

```
string imap_clearflag_full  (int stream, string sequence, string flag, string options)
```

This function causes a store to delete the specified flag to the flags set for the messages in the specified sequence. The flags which you can unset are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", and "\\Recent" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_close (PHP 3, PHP 4)

Close an IMAP stream

```
int imap_close (int imap_stream [, int flags])
```

Close the imap stream. Takes an optional *flag* CL_EXPUNGE, which will silently expunge the mailbox before closing, removing all messages marked for deletion.

imap_createmailbox (PHP 3, PHP 4)

Create a new mailbox

```
int imap_createmailbox (int imap_stream, string mbox)
```

imap_createmailbox() creates a new mailbox specified by *mbox*. Names containing international characters should be encoded by **imap_utf7_encode()**

Returns true on success and false on error.

See also **imap_renamemailbox()**, **imap_deletemailbox()** and **imap_open()** for the format of *mbox* names.

Příklad 1. imap_createmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
     || die("can't connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewböx");
```

```
$newname = $name1;

echo "Newname will be ’$name1’<br>\n";

# we will now create a new mailbox "phptestbox" in your inbox folder,
# check its status after creation and finaly remove it to restore
# your inbox to its initial state
if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname"))) {
  $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
  if($status) {
    print("your new mailbox ’$name1’ has the following status:<br>\n");
    print("Messages:   ". $status->messages   ."<br>\n");
    print("Recent:     ". $status->recent     ."<br>\n");
    print("Unseen:     ". $status->unseen     ."<br>\n");
    print("UIDnext:    ". $status->uidnext    ."<br>\n");
    print("UIDvalidity:". $status->uidvalidity."<br>\n");

    if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$name2"
      echo "renamed new mailbox from ’$name1’ to ’$name2’<br>\n";
      $newname=$name2;
    } else {
      print "imap_renamemailbox on new mailbox failed: ".imap_last_error()."<br>\n";
    }
  } else {
    print  "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
  }
  if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
    print "new mailbox removed to restore initial state<br>\n";
  } else {
    print  "imap_deletemailbox on new mail-
box failed: ".implode("<br>\n",imap_errors())."<br>\n";
  }

} else {
  print  "could not create new mailbox: ".implode("<br>\n",imap_errors())."<br>\n";
}

imap_close($mbox);
```

imap_delete (PHP 3, PHP 4)

Mark a messge for deletion from current mailbox

```
int imap_delete (int imap_stream, int msg_number [, int flags])
```

Returns true.

imap_delete() function marks message pointed by *msg_number* for deletion. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. Messages marked for deletion will stay in the mailbox until either **imap_expunge()** is called or **imap_close()** is called with the optional parameter CL_EXPUNGE.

Příklad 1. Imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    || die ("can’t connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
print "Messages before delete: " . $check->Nmsgs . "<br>\n" ;
```

```
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after  delete: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after expunge: " . $check->Nmsgs . "<br>\n" ;
imap_close ($mbox);
```

imap_deletemailbox (PHP 3, PHP 4)

Delete a mailbox

```
int imap_deletemailbox (int imap_stream, string mbox)
```

imap_deletemailbox() deletes the specified mailbox (see **imap_open()** for the format of *mbox* names).

Returns true on success and false on error.

See also **imap_createmailbox()**, **imap_renamemailbox()**, and **imap_open()** for the format of *mbox*.

imap_errors (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

This function returns all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.

```
array imap_errors (void)
```

This function returns an array of all of the IMAP error messages generated since the last **imap_errors()** call, or the beginning of the page. When **imap_errors()** is called, the error stack is subsequently cleared.

imap_expunge (PHP 3, PHP 4)

Delete all messages marked for deletion

```
int imap_expunge (int imap_stream)
```

imap_expunge() deletes all the messages marked for deletion by **imap_delete()**, **imap_mail_move()**, or **imap_setflag_full()**.

Returns true.

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4)

Read an overview of the information in the headers of the given message

```
array imap_fetch_overview (int imap_stream, string sequence [, int flags])
```

This function fetches mail headers for the given *sequence* and returns an overview of their contents. *sequence* will contain a sequence of message indices or UIDs, if *flags* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the maibox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Příklad 1. imap_fetch_overview() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     || die("can't connect: ".imap_last_error());

$overview = imap_fetch_overview($mbox,"2,4:6",0);

if(is_array($overview)) {
        reset($overview);
        while( list($key,$val) = each($overview)) {
                print     $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
}

imap_close($mbox);
```

imap_fetchbody (PHP 3, PHP 4)

Fetch a particular section of the body of the message

```
string imap_fetchbody (int imap_stream, int msg_number, string part_number [, flags
flags])
```

This function causes a fetch of a particular section of the body of the specified messages as a text string and returns that text string. The section specification is a string of integers delimited by period which index into a body part list as per the IMAP4 specification. Body parts are not decoded by this function.

The options for **imap_fetchbody()** is a bitmask with one or more of the following:

- FT_UID - The *msg_number* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set

- FT_INTERNAL - The return string is in "internal" format, without any attempt to canonicalize CRLF.

See also: **imap_fetchstructure()**.

imap_fetchheader (PHP 3>= 3.0.3, PHP 4)

Returns header for a message

```
string imap_fetchheader (int imap_stream, int msgno, int flags)
```

This function causes a fetch of the complete, unfiltered RFC822 (http://www.faqs.org/rfcs/rfc822.html) format header of the specified message as a text string and returns that text string.

The options are:

FT_UID The msgno argument is a UID
FT_INTERNAL The return string is in "internal" format, without any attempt to canonicalize to CRLF newlines
FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the same time. This avoids an extra RTT on an IMAP connection if a full message text is desired (e.g. in a "save to local file" operation)

imap_fetchstructure (PHP 3, PHP 4)

Read the structure of a particular message

```
object imap_fetchstructure  (int imap_stream, int msg_number [, int flags])
```

This function fetches all the structured information for a given message. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. The returned object includes the envelope, internal date, size, flags and body structure along with a similar object for each mime attachement. The structure of the returned objects is as follows:

Tabulka 1. Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	True if there is a subtype string
subtype	MIME subtype
ifdescription	True if there is a description string
description	Content description string
ifid	True if there is an identification string
id	Identification string
lines	Number of lines
bytes	Number of bytes
ifdisposition	True if there is a disposition string

disposition	Disposition string
ifdparameters	True if the dparameters array exists
dparameters	Disposition parameter array
ifparameters	True if the parameters array exists
parameters	MIME parameters array
parts	Array of objects describing each message part

1. dparameters is an array of objects where each object has an "attribute" and a "value" property.
2. Parameter is an array of objects where each object has an "attributte" and a "value" property.
3. Parts is an array of objects identical in structure to the top-level object, with the limitation that it cannot contain further 'parts' objects.

Tabulka 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Tabulka 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

See also: **imap_fetchbody()**.

imap_get_quota (PHP 4 CVS only)

Retrieve the quota level settings, and usage statics per mailbox

```
array imap_get_quota (int imap_stream, string quota_root)
```

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return FALSE in the case of failure.

This function is currently only available to users of the c-client2000 library.

imap_stream should be the value returned from an **imap_status()** call. This stream is required to be opened as the mail admin user for the quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Příklad 1. imap_get_quota() example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      || die("can't connect: ".imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_value)) {
    print "Usage level is: " . $quota_value['usage'];
    print "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
```

See also **imap_open()**, **imap_set_quota()**.

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Read the list of mailboxes, returning detailed information on each one

```
array imap_getmailboxes (int imap_stream, string ref, string pattern)
```

Returns an array of objects containing mailbox information. Each object has the attributes *name*, specifying the full name of the mailbox; *delimiter*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *attributes*. *Attributes* is a bitmask that can be tested against:

- LATT_NOINFERIORS - This mailbox has no "children" (there are no mailboxes below this one).
- LATT_NOSELECT - This is only a container, not a mailbox - you cannot open it.
- LATT_MARKED - This mailbox is marked. Only used by UW-IMAPD.
- LATT_UNMARKED - This mailbox is not marked. Only used by UW-IMAPD.

Mailbox names containing international Characters outside the printable ASCII range will be encoded and may be decoded by **imap_utf7_decode()**.

ref should normally be just the server specification as described in **imap_open()**, and *pattern* specifies where in the mailbox hierarchy to start searching. If you want all mailboxes, pass '*' for *pattern*.

There are two special characters you can pass as part of the *pattern*: '*' and '%'. '*' means to return all mailboxes. If you pass *pattern* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *pattern* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Příklad 1. imap_getmailboxes() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      || die("can't connect: ".imap_last_error());

$list = imap_getmailboxes($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
  {
    print "($key) ";
```

```
    print imap_utf7_decode($val->name).",";
    print "'".$val->delimiter."',";
    print $val->attributes."<br>\n";
  }
} else
  print "imap_getmailboxes failed: ".imap_last_error()."\n";

imap_close($mbox);
```

See also **imap_getsubscribed()**.

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

List all the subscribed mailboxes

```
array imap_getsubscribed (int imap_stream, string ref, string pattern)
```

This function is identical to **imap_getmailboxes()**, except that it only returns mailboxes that the user is subscribed to.

imap_header (PHP 3, PHP 4)

Read the header of the message

```
object imap_header (int imap_stream, int msg_number [, int fromlength [, int
subjectlength [, string defaulthost]]])
```

This is an alias to **imap_headerinfo()** and is identical to this in any way.

imap_headerinfo (PHP 3, PHP 4)

Read the header of the message

```
object imap_headerinfo (int imap_stream, int msg_number [, int fromlength [, int
subjectlength [, string defaulthost]]])
```

This function returns an object of various header elements.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' if recent and seen,
'N' if recent and not seen,
' ' if not recent
Unseen - 'U' if not seen AND not recent,
' ' if seen OR not seen and recent
Answered -'A' if answered,
' ' if unanswered
Deleted - 'D' if deleted,
' ' if not deleted
Draft - 'X' if draft,
' ' if not draft

Flagged - 'F' if flagged,
' ' if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to know if a message is Unseen, you must check for

Unseen == 'U' || Recent == 'N'

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
personal
adl
mailbox
host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
personal
adl
mailbox
host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing):
personal
adl
mailbox
host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
personal
adl
mailbox
host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line, containing):
personal
adl
mailbox
host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
personal
adl
mailbox
host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line, containing):
personal
adl
mailbox
host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength* characters)

fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers (PHP 3, PHP 4)

Returns headers for all messages in a mailbox

```
array imap_headers (int imap_stream)
```

Returns an array of string formatted with header info. One element per mail message.

imap_last_error (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

This function returns the last IMAP error (if any) that occurred during this page request

```
string imap_last_error (void)
```

This function returns the full text of the last IMAP error message that occurred on the current page. The error stack is untouched; calling **imap_last_error()** subsequently, with no intervening errors, will return the same error.

imap_listmailbox (PHP 3, PHP 4)

Read the list of mailboxes

```
array imap_listmailbox (int imap_stream, string ref, string pattern)
```

Returns an array containing the names of the mailboxes. See **imap_getmailboxes()** for a description of *ref* and *pattern*.

Příklad 1. imap_listmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      || die("can't connect: ".imap_last_error());

$list = imap_listmailbox($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
    print imap_utf7_decode($val)."<br>\n";
} else
  print "imap_listmailbox failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_listsubscribed (PHP 3, PHP 4)

List all the subscribed mailboxes

```
array imap_listsubscribed (int imap_stream, string ref, string pattern)
```

Returns an array of all the mailboxes that you have subscribed. This is almost identical to **imap_listmailbox()**, but will only return mailboxes the user you logged in as has subscribed.

imap_mail (PHP 3>= 3.0.14, PHP 4 >= 4.0b4)

Send an email message

```
string imap_mail (string to, string subject, string message [, string additional_headers
[, string cc [, string bcc [, string rpath]]]])
```

This function is currently only available in PHP 3.

imap_mail_compose (PHP 3>= 3.0.5, PHP 4)

Create a MIME message based on given envelope and body sections

```
string imap_mail_compose (array envelope, array body)
```

Příklad 1. imap_mail_compose() example

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));
```

```
?>
```

imap_mail_copy (PHP 3, PHP 4)

Copy specified messages to a mailbox

```
int imap_mail_copy (int imap_stream, string msglist, string mbox [, int flags])
```

Returns true on success and false on error.

Copies mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask of one or more of

- CP_UID - the sequence numbers contain UIDS
- CP_MOVE - Delete the messages from the current mailbox after copying

imap_mail_move (PHP 3, PHP 4)

Move specified messages to a mailbox

```
int imap_mail_move (int imap_stream, string msglist, string mbox [, int flags])
```

Moves mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask and may contain the single option

- CP_UID - the sequence numbers contain UIDS

Returns true on success and false on error.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4)

Get information about the current mailbox

```
object imap_mailboxmsginfo (int imap_stream)
```

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_mailboxmsginfo()** function checks the current mailbox status on the server. It is similar to **imap_status()**, but will additionally sum up the size of all messages in the mailbox, which will take some additional time to execute. It returns the information in an object with following properties.

Tabulka 1. Mailbox properties

Date	date of last change

Driver	driver
Mailbox	name of the mailbox
Nmsgs	number of messages
Recent	number of recent messages
Unread	number of unread messages
Deleted	number of deleted messages
Size	mailbox size

Příklad 1. imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{your.imap.host}INBOX","username", "password")
      || die("can't connect: ".imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if($check) {
    print "Date: "    . $check->Date    ."<br>\n" ;
    print "Driver: "  . $check->Driver  ."<br>\n" ;
    print "Mailbox: " . $check->Mailbox ."<br>\n" ;
    print "Messages: ". $check->Nmsgs   ."<br>\n" ;
    print "Recent: "  . $check->Recent  ."<br>\n" ;
    print "Unread: "  . $check->Unread  ."<br>\n" ;
    print "Deleted: " . $check->Deleted ."<br>\n" ;
    print "Size: "    . $check->Size    ."<br>\n" ;
} else {
    print "imap_check() failed: ".imap_last_error(). "<br>\n";
}

imap_close($mbox);

?>
```

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4 >= 4.0RC1)

Decode MIME header elements

```
array imap_header_decode (string text)
```

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047 (http://www.faqs.org/rfcs/rfc2047.html)) The decoded elements are returned in an array of objects, where each object has two properties, "charset" & "text". If the element hasn't been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Příklad 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";

$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
echo "Charset: {$elements[$i]->charset}\n";
echo "Text: {$elements[$i]->text}\n\n";
```

```
}
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno (PHP 3>= 3.0.3, PHP 4)

This function returns the message sequence number for the given UID

```
int imap_msgno (int imap_stream, int uid)
```

This function returns the message sequence number for the given UID. It is the inverse of **imap_uid()**.

imap_num_msg (PHP 3, PHP 4)

Gives the number of messages in the current mailbox

```
int imap_num_msg (int imap_stream)
```

Return the number of messages in the current mailbox.

See also: **imap_num_recent()** and **imap_status()**.

imap_num_recent (PHP 3, PHP 4)

Gives the number of recent messages in current mailbox

```
int imap_num_recent (int imap_stream)
```

Returns the number of recent messages in the current mailbox.

See also: **imap_num_msg()** and **imap_status()**.

imap_open (PHP 3, PHP 4)

Open an IMAP stream to a mailbox

```
int imap_open (string mailbox, string username, string password [, int flags])
```

Returns an IMAP stream on success and false on error. This function can also be used to open streams to POP3 and NNTP servers, but some functions and features are not available on IMAP servers.

A mailbox name consists of a server part and a mailbox path on this server. The special name INBOX stands for the current users personal mailbox. The server part, which is enclosed in '{' and '}', consists of the servers name or ip address, a protocol secification (beginning with '/') and an optional port specifier beginnung with ':'. The server part is mandatory in all mailbox parameters. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded with **imap_utf7_encode()**.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

To connect to an IMAP server running on port 143 on the local machine, do the following:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

To connect to a POP3 server on port 110 on the local server, use:

```
$mbox = imap_open ("{localhost/pop3:110}INBOX", "user_id", "password");
```

To connect to an NNTP server on port 119 on the local server, use:

```
$nntp = imap_open ("{localhost/nntp:119}comp.test", "", "");
```

To connect to a remote server replace "localhost" with the name or the IP address of the server you want to connect to.

Příklad 1. imap_open() example

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");

if ($folders == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key, $val) = each ($folders)) {
        echo $val."<br>\n";
    }
}

echo "<p><h1>Headers in INBOX</h1>\n";
$headers = imap_headers ($mbox);

if ($headers == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key,$val) = each ($headers)) {
        echo $val."<br>\n";
    }
}

imap_close($mbox);
```

imap_ping (PHP 3, PHP 4)

Check if the IMAP stream is still active

```
int imap_ping (int imap_stream)
```

Returns true if the stream is still alive, false otherwise.

imap_ping() function pings the stream to see it is still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout. (As PHP scripts do not tend to run that long, i can hardly imagine that this function will be usefull to anyone.)

imap_qprint (PHP 3, PHP 4)

Convert a quoted-printable string to an 8 bit string

```
string imap_qprint (string string)
```

Convert a quoted-printable string to an 8 bit string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns an 8 bit (binary) string.

See also **imap_8bit()**.

imap_renamemailbox (PHP 3, PHP 4)

Rename an old mailbox to new mailbox

```
int imap_renamemailbox (int imap_stream, string old_mbox, string new_mbox)
```

This function renames on old mailbox to new mailbox (see **imap_open()** for the format of *mbox* names).

Returns true on success and false on error.

See also **imap_createmailbox()**, **imap_deletemailbox()**, and **imap_open()** for the format of *mbox*.

imap_reopen (PHP 3, PHP 4)

Reopen IMAP stream to new mailbox

```
int imap_reopen (int imap_stream, string mailbox [, string flags])
```

This function reopens the specified stream to a new mailbox on an IMAP or NNTP server.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox.
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close (see also **imap_delete()** and **imap_expunge()**)

Returns true on success and false on error.

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4)

Parses an address string

```
array imap_rfc822_parse_adrlist  (string address, string default_host)
```

This function parses the address string as defined in RFC822 (http://www.faqs.org/rfcs/rfc822.html) and for each address, returns an array of objects. The objects properties are:

- mailbox - the mailbox name (username)
- host - the host name
- personal - the personal name
- adl - at domain source route

Příklad 1. imap_rfc822_parse_adrlist() example

```
$address_string = "Hartmut Holzgraefe <hartmut@cvs.php.net>, postmas-
ter@somedomain.net, root";
$address_array  = imap_rfc822_parse_adrlist($address_string,"somedomain.net");
if(! is_array($address_array)) die("somethings wrong\n");

reset($address_array);
while(list($key,$val)=each($address_array)){
  print "mailbox : ".$val->mailbox."<br>\n";
  print "host    : ".$val->host."<br>\n";
  print "personal: ".$val->personal."<br>\n";
  print "adl     : ".$val->adl."<p>\n";
}
```

imap_rfc822_parse_headers (PHP 4 >= 4.0RC1)

Parse mail headers from a string

```
object imap_rfc822_parse_headers (string headers [, string defaulthost])
```

This function returns an object of various header elements, similar to **imap_header()**, except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4)

Returns a properly formatted email address given the mailbox, host, and personal info.

```
string imap_rfc822_write_address  (string mailbox, string host, string personal)
```

Returns a properly formatted email address as defined in RFC822 (http://www.faqs.org/rfcs/rfc822.html) given the mailbox, host, and personal info.

Příklad 1. imap_rfc822_write_address() example

```
print imap_rfc822_write_address("hartmut","cvs.php.net","Hartmut Holzgraefe")."\n";
```

imap_scanmailbox (PHP 3, PHP 4)

Read the list of mailboxes, takes a string to search for in the text of the mailbox

```
array imap_scanmailbox (int imap_stream, string ref, string pattern, string content)
```

Returns an array containing the names of the mailboxes that have *string* in the text of the mailbox. This function is similar to **imap_listmailbox()**, but it will additionally check for the presence of the string *content* inside the mailbox data. See **imap_getmailboxes()** for a description of *ref* and *pattern*.

imap_search (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

This function returns an array of messages matching the given search criteria

```
array imap_search (int imap_stream, string criteria, int flags)
```

This function performs a search on the mailbox currently opened in the given imap stream. *criteria* is a string, delimited by spaces, in which the following keywords are allowed. Any multi-word arguments (eg. FROM "joey smith") must be quoted.

- ALL - return all messages matching the rest of the criteria
- ANSWERED - match messages with the \\ANSWERED flag set
- BCC "string" - match messages with "string" in the Bcc: field
- BEFORE "date" - match messages with Date: before "date"
- BODY "string" - match messages with "string" in the body of the message
- CC "string" - match messages with "string" in the Cc: field
- DELETED - match deleted messages
- FLAGGED - match messages with the \\FLAGGED (sometimes referred to as Important or Urgent) flag set
- FROM "string" - match messages with "string" in the From: field
- KEYWORD "string" - match messages with "string" as a keyword
- NEW - match new messages
- OLD - match old messages
- ON "date" - match messages with Date: matching "date"
- RECENT - match messages with the \\RECENT flag set
- SEEN - match messages that have been read (the \\SEEN flag is set)
- SINCE "date" - match messages with Date: after "date"
- SUBJECT "string" - match messages with "string" in the Subject:
- TEXT "string" - match messages with text "string"
- TO "string" - match messages with "string" in the To:
- UNANSWERED - match messages that have not been answered
- UNDELETED - match messages that are not deleted
- UNFLAGGED - match messages that are not flagged
- UNKEYWORD "string" - match messages that do not have the keyword "string"
- UNSEEN - match messages which have not been read yet

For example, to match all unanswered messages sent by Mom, you'd use: "UNANSWERED FROM mom". Searches appear to be case insensitive. This list of criteria is from a reading of the UW c-client source code and may be uncomplete or inaccurate (see also RFC2060, section 6.4.4).

Valid values for flags are SE_UID, which causes the returned array to contain UIDs instead of messages sequence numbers.

imap_set_quota (PHP 4 CVS only)

Sets a quota for a given mailbox

```
int imap_set_quota (int imap_stream, string quota_root, int quota_limit)
```

Sets an upper limit quota on a per mailbox basis. This function requires the *imap_stream* to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 library.

imap_stream is the stream pointer returned from a **imap_open()** call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns true on success and false on error.

Příklad 1. imap_set_quota() example

```
$mbox = imap_open ("{your.imap.host:143}", "mailadmin", "password");

if(!imap_set_quota($mbox, "user.kalowsky", 3000)) {
print "Error in setting quota\n";
return;
}

imap_close($mbox);
```

See also **imap_open()**, **imap_set_quota()**.

imap_setflag_full (PHP 3>= 3.0.3, PHP 4)

Sets flags on messages

```
string imap_setflag_full (int stream, string sequence, string flag, string options)
```

This function causes a store to add the specified flag to the flags set for the messages in the specified sequence.

The flags which you can set are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", and "\\Recent" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

Příklad 1. imap_setflag_full() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     || die("can't connect: ".imap_last_error());

$status = imap_setflag_full($mbox,"2,5","\\Seen \\Flagged");

print gettype($status)."\n";
print $status."\n";

imap_close($mbox);
```

imap_sort (PHP 3>= 3.0.3, PHP 4)

Sort an array of message headers

```
array imap_sort (int stream, int criteria, int reverse, int options)
```

Returns an array of message numbers sorted by the given parameters.

Reverse is 1 for reverse-sorting.

Criteria can be one (and only one) of the following:

SORTDATE message Date
SORTARRIVAL arrival date
SORTFROM mailbox in first From address
SORTSUBJECT message Subject
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE size of message in octets

The flags are a bitmask of one or more of the following:

SE_UID Return UIDs instead of sequence numbers
SE_NOPREFETCH Don't prefetch searched messages.

imap_status (PHP 3>= 3.0.4, PHP 4)

This function returns status information on a mailbox other than the current one

```
object imap_status (int imap_stream, string mailbox, int options)
```

This function returns an object containing status information. Valid flags are:

- SA_MESSAGES - set status->messages to the number of messages in the mailbox
- SA_RECENT - set status->recent to the number of recent messages in the mailbox
- SA_UNSEEN - set status->unseen to the number of unseen (new) messages in the mailbox
- SA_UIDNEXT - set status->uidnext to the next uid to be used in the mailbox

- SA_UIDVALIDITY - set status->uidvalidity to a constant that changes when uids for the mailbox may no longer be valid
- SA_ALL - set all of the above

status->flags is also set, which contains a bitmask which can be checked against any of the above constants.

Příklad 1. imap_status() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      || die("can't connect: ".imap_last_error());

$status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
if($status) {
  print("Messages:   ". $status->messages   ."<br>\n";
  print("Recent:     ". $status->recent     ."<br>\n";
  print("Unseen:     ". $status->unseen     ."<br>\n";
  print("UIDnext:    ". $status->uidnext    ."<br>\n";
  print("UIDvalidity:". $status->uidvalidity."<br>\n";
} else
  print "imap_status failed: ".imap_lasterror()."\n";

imap_close($mbox);
```

imap_subscribe (PHP 3, PHP 4)

Subscribe to a mailbox

```
int imap_subscribe (int imap_stream, string mbox)
```

Subscribe to a new mailbox.

Returns true on success and false on error.

imap_uid (PHP 3>= 3.0.3, PHP 4)

This function returns the UID for the given message sequence number

```
int imap_uid (int imap_stream, int msgno)
```

This function returns the UID for the given message sequence number. An UID is an unique identifier that will not change over time while a message sequence number may change whenever the content of the mailbox changes. This function is the inverse of **imap_msgno()**.

imap_undelete (PHP 3, PHP 4)

Unmark the message which is marked deleted

```
int imap_undelete (int imap_stream, int msg_number)
```

This function removes the deletion flag for a specified message, which is set by **imap_delete()** or **imap_mail_move()**.

Returns true on success and false on error.

imap_unsubscribe (PHP 3, PHP 4)

Unsubscribe from a mailbox

```
int imap_unsubscribe (int imap_stream, string mbox)
```

Unsubscribe from a specified mailbox.

Returns true on success and false on error.

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4 >= 4.0b4)

Decodes a modified UTF-7 encoded string.

```
string imap_utf7_decode (string text)
```

Decodes modified UTF-7 *text* into 8bit data.

Returns the decoded 8bit data, or false if the input string was not valid modified UTF-7. This function is needed to decode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4 >= 4.0b4)

Converts 8bit data to modified UTF-7 text.

```
string imap_utf7_encode (string data)
```

Converts 8bit *data* to modified UTF-7 text. This is needed to encode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

Returns the modified UTF-7 text.

imap_utf8 (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Converts text to UTF8

```
string imap_utf8 (string text)
```

Converts the given *text* to UTF8 (as defined in RFC2044 (http://www.faqs.org/rfcs/rfc2044.html)).

XXXIV. Informix functions

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php3_ifx.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Configuration notes: You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK.

Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

The configure script will autodetect the libraries and include directories, if you run "configure –with_informix=yes". You can overide this detection by specifying "IFX_LIBDIR", "IFX_LIBS" and "IFX_INCDIR" in the environment. The configure script will also try to detect your Informix server version. It will set the "HAVE_IFX_IUS" conditional compilation variable if your Informix version >= 9.00.

Runtime considerations: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to **phpinfo()()** before you start testing. The **phpinfo()()** output should list these environment variables. This is true for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LINBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with : "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "**ifx_create_blob()**;". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with **ifx_update_blob()**.

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime :

configuration variable : ifx.textasvarchar

configuration variable : ifx.byteasvarchar

runtime functions :

ifx_textasvarchar(0) : use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0) : use blob id's for select queries with BYTE columns

ifx_textasvarchar(1) : return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1) : return BYTE columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

configuration variable : ifx.blobinfile

runtime function :

ifx_blobinfile_mode(0) : return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1) : return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like : putenv(blobdir=tmpblob"); will ease the cleaning up of temp files accidentally

left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar : if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

NULL values: The configuration variable ifx.nullformat (and the runtime function **ifx_nullformat()**) when set to true will return NULL columns as the string "NULL", when set to false they return the empty string. This allows you to discriminate between NULL columns and empty columns.

ifx_connect (PHP 3>= 3.0.3, PHP 4)

Open Informix server connection

```
int ifx_connect ([string database [, string userid [, string password]]])
```

Returns a connection identifier on success, or FALSE on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in configuration file (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **ifx_close()**.

See also **ifx_pconnect()**, and **ifx_close()**.

Příklad 1. Connect to a Informix database

```
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
```

ifx_pconnect (PHP 3>= 3.0.3, PHP 4)

Open persistent Informix connection

```
int ifx_pconnect ([string database [, string userid [, string password]]])
```

Returns: A positive Informix persistent link identifier on success, or false on error

ifx_pconnect() acts very much like **ifx_connect()** with two major differences.

This function behaves exactly like **ifx_connect()** when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ifx_close()** will not close links established by **ifx_pconnect()**).

This type of links is therefore called 'persistent'.

See also: **ifx_connect()**.

ifx_close (PHP 3>= 3.0.3, PHP 4)

Close Informix connection

```
int ifx_close ([int link_identifier])
```

Returns: always true.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by **ifx_pconnect()**.

See also: **ifx_connect()**, and **ifx_pconnect()**.

Příklad 1. Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_query (PHP 3>= 3.0.3, PHP 4)

Send Informix query

```
int ifx_query (string query [, int link_identifier [, int cursor_type [, mixed
blobidarray]]])
```

Returns: A positive Informix result identifier on success, or false on error.

A "result_id" resource used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the **ifx_affected_rows()** function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **ifx_connect()** was called, and use it.

Executes *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate". IFX_SCROLL and IFX_HOLD are symbolic constants and as such shouldn't be between quotes. I you omit this parameter the cursor is a normal sequential cursor.

For either query type the number of (estimated or real) affected rows is saved for retrieval by **ifx_affected_rows()**.

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: **ifx_connect()**.

Příklad 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Příklad 2. Insert some values into the "catalog" table

```
                      // create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
```

```
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                    // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                     // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                    // free result id
ifx_free_result($res_id);
```

ifx_prepare (PHP 3>= 3.0.4, PHP 4)

Prepare an SQL-statement for execution

```
int ifx_prepare (string query, int conn_id [, int cursor_def, mixed blobidarray])
```

Returns a integer *result_id* for use by **ifx_do()**. Sets *affected_rows* for retrieval by the **ifx_affected_rows()** function.

Prepares *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together.

For either query type the estimated number of affected rows is saved for retrieval by **ifx_affected_rows()**.

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: **ifx_do()**.

ifx_do (PHP 3>= 3.0.4, PHP 4)

Execute a previously prepared SQL-statement

```
int ifx_do (int result_id)
```

Returns TRUE on success, FALSE on error.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *result_id* on error.

Also sets the real number of **ifx_affected_rows()** for non-select statements for retrieval by **ifx_affected_rows()**

See also: **ifx_prepare()**. There is a example.

ifx_error (PHP 3>= 3.0.3, PHP 4)

Returns error code of last Informix call

```
string ifx_error(void);
```

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be ' ' (space) (success), 'W' (the statement caused some warning), 'E' (an error happened when executing the statement) or 'N' (the statement didn't return any data).

See also: **ifx_errormsg()**

ifx_errormsg (PHP 3>= 3.0.4, PHP 4)

Returns error message of last Informix call

```
string ifx_errormsg ([int errorcode])
```

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*errorcode*" param is present, the error message corresponding to "*errorcode*".

See also: **ifx_error()**

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4)

Get number of rows affected by a query

```
int ifx_affected_rows (int result_id)
```

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns the number of rows affected by a query associated with *result_id*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after **ifx_prepare()** to limit queries to reasonable result sets.

See also: **ifx_num_rows()**

Příklad 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4)

Get the contents of sqlca.sqlerrd[0..5] after a query

```
array ifx_getsqlca (int result_id)
```

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns a pseudo-row (assiociative array) with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For SELECTs the values are those saved after the PREPARE statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Příklad 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4)

Get row as enumerated array

```
array ifx_fetch_row (int result_id [, mixed position])
```

Returns an associative array that corresponds to the fetched row, or false if there are no more rows.

Blob columns are returned as integer blob id values for use in **ifx_get_blob()** unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns FALSE on error

result_id is a valid resultid returned by **ifx_query()** or **ifx_prepare()** (select type queries only!).

position is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0, with the column name as key.

Subsequent calls to **ifx_fetch_row()** would return the next row in the result set, or false if there are no more rows.

Příklad 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4)

Formats all rows of a query into a HTML table

```
int ifx_htmltbl_result (int result_id [, string html_table_options])
```

Returns the number of rows fetched or FALSE on error.

Formats all rows of the *result_id* query into a html table. The optional second argument is a string of <table> tag options

Příklad 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4)

List of Informix SQL fields

```
array ifx_fieldtypes (int result_id)
```

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *result_id*. Returns FALSE on error.

Příklad 1. Fielnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4)

List of SQL fieldproperties

```
array ifx_fieldproperties (int result_id)
```

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns FALSE on error.

Returns the Informix SQL fieldproperies of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Příklad 1. Informix SQL fieldproperties

```
$properties = ifx_fieldproperties ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_num_fields (PHP 3>= 3.0.3, PHP 4)

Returns the number of columns in the query

```
int ifx_num_fields (int result_id)
```

Returns the number of columns in query for *result_id* or FALSE on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4)

Count the rows already fetched from a query

```
int ifx_num_rows (int result_id)
```

Gives the number of rows fetched so far for a query with *result_id* after a **ifx_query()** or **ifx_do()** query.

ifx_free_result (PHP 3>= 3.0.3, PHP 4)

Releases resources for the query

```
int ifx_free_result (int result_id)
```

Releases resources for the query associated with *result_id*. Returns FALSE on error.

ifx_create_char (PHP 3>= 3.0.6, PHP 4)

Creates an char object

```
int ifx_create_char (string param)
```

Creates an char object. *param* should be the char content.

ifx_free_char (PHP 3>= 3.0.6, PHP 4)

Deletes the char object

```
int ifx_free_char (int bid)
```

Deletes the charobject for the given char object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_update_char (PHP 3>= 3.0.6, PHP 4)

Updates the content of the char object

```
int ifx_update_char (int bid, string content)
```

Updates the content of the char object for the given char object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_get_char (PHP 3>= 3.0.6, PHP 4)

Return the content of the char object

```
int ifx_get_char (int bid)
```

Returns the content of the char object for the given char object-id *bid*.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4)

Creates an blob object

```
int ifx_create_blob (int type, int mode, string param)
```

Creates an blob object.
type: 1 = TEXT, 0 = BYTE
mode: 0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.
param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.
Return FALSE on error, otherwise the new blob object-id.

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4)

Duplicates the given blob object

```
int ifx_copy_blob (int bid)
```

Duplicates the given blob object. *bid* is the ID of the blob object.
Returns FALSE on error otherwise the new blob object-id.

ifx_free_blob (PHP 3>= 3.0.4, PHP 4)

Deletes the blob object

```
int ifx_free_blob (int bid)
```

Deletes the blobobject for the given blob object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4)

Return the content of a blob object

```
int ifx_get_blob (int bid)
```

Returns the content of the blob object for the given blob object-id *bid*.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4)

Updates the content of the blob object

```
ifx_update_blob (int bid, string content)
```

Updates the content of the blob object for the given blob object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4)

Set the default blob mode for all select queries

```
void ifx_blobinfile_mode (int mode)
```

Set the default blob mode for all select queries. Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4)

Set the default text mode

```
void ifx_textasvarchar (int mode)
```

Sets the default text mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4)

Set the default byte mode

```
void ifx_byteasvarchar (int mode)
```

Sets the default byte mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_nullformat (PHP 3>= 3.0.4, PHP 4)

Sets the default return value on a fetch row

```
void ifx_nullformat (int mode)
```

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "NULL".

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4)

Creates an slob object and opens it

```
int ifxus_create_slob (int mode)
```

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return FALSE on error otherwise the new slob object-id.

ifxus_free_slob (PHP 3>= 3.0.4, PHP 4)

Deletes the slob object

```
int ifxus_free_slob (int bid)
```

Deletes the slob object. *bid* is the Id of the slob object. Returns FALSE on error otherwise TRUE.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4)

Deletes the slob object

```
int ifxus_close_slob (int bid)
```

Deletes the slob object on the given slob object-id *bid*. Return FALSE on error otherwise TRUE.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4)

Opens an slob object

```
int ifxus_open_slob (long bid, int mode)
```

Opens an slob object. *bid* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. Returns FALSE on error otherwise the new slob object-id.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4)

Returns the current file or seek position

```
int ifxus_tell_slob (long bid)
```

Returns the current file or seek position of an open slob object *bid* should be an existing slob id. Return FALSE on error otherwise the seek position.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4)

Sets the current file or seek position

```
int ifxus_seek_blob (long bid, int mode, long offset)
```

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return FALSE on error otherwise the seek position.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4)

Reads nbytes of the slob object

```
int ifxus_read_slob (long bid, long nbytes)
```

Reads nbytes of the slob object. *`bid`* is a existing slob id and *`nbytes`* is the number of bytes zu read. Return FALSE on error otherwise the string.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4)

Writes a string into the slob object

```
int ifxus_write_slob (long bid, string content)
```

Writes a string into the slob object. *`bid`* is a existing slob id and *`content`* the content to write. Return FALSE on error otherwise bytes written.

XXXV. InterBase functions

InterBase is a popular database put out by Borland/Inprise. More information about InterBase is available at http://www.interbase.com/. Oh, by the way, InterBase just joined the open source movement!

Poznámka: Full support for InterBase 6 was added in PHP 4.0.

This database uses a single quote (') character for escaping, a behavior similar to the Sybase database, add to your `php.ini` the following directive:

```
magic_quotes_sybase = On
```

ibase_connect (PHP 3>= 3.0.6, PHP 4)

Open a connection to an InterBase database

```
int ibase_connect (string database [, string username [, string password [, string
charset [, int buffers [, int dialect [, string role]]]]]])
```

Establishes a connection to an InterBase server. The *database* argument has to be a valid path to database file on the server it resides on. If the server is not local, it must be prefixed with either 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) or 'hostname@' (IPX/SPX), depending on the connection protocol used. *username* and *password* can also be specified with PHP configuration directives ibase.default_user and ibase.default_password. *charset* is the default character set for a database. *buffers* is the number of database buffers to allocate for the server-side cache. If 0 or omitted, server chooses its own default. *dialect* selects the default SQL dialect for any statement executed within a connection, and it defaults to the highest one supported by client libraries.

In case a second call is made to **ibase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **ibase_close()**.

Příklad 1. Ibase_connect() example

```
<?php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Poznámka: *buffers* was added in PHP4-RC2.

Poznámka: *dialect* was added in PHP4-RC2. It is functional only with InterBase 6 and versions higher than that.

Poznámka: *role* was added in PHP4-RC2. It is functional only with InterBase 5 and versions higher than that.

See also: **ibase_pconnect()**.

ibase_pconnect (PHP 3>= 3.0.6, PHP 4)

Creates an persistent connection to an InterBase database

```
int ibase_pconnect (string database [, string username [, string password [, string
charset [, int buffers [, int dialect [, string role]]]]]])
```

ibase_pconnect() acts very much like **ibase_connect()** with two major differences. First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the InterBase server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ibase_close()** will not close links established by **ibase_pconnect()**). This type of link is therefore called 'persistent'.

Poznámka: *buffers* was added in PHP4-RC2.

Poznámka: *dialect* was added in PHP4-RC2. It is functional only with InterBase 6 and versions higher than that.

Poznámka: *role* was added in PHP4-RC2. It is functional only with InterBase 5 and versions higher than that.

See also **ibase_connect()** for the meaning of parameters passed to this function. They are exactly the same.

ibase_close (PHP 3>= 3.0.6, PHP 4)

Close a connection to an InterBase database

```
int ibase_close ([int connection_id])
```

Closes the link to an InterBase database that's associated with a connection id returned from **ibase_connect()**. If the connection id is omitted, the last opened link is assumed. Default transaction on link is committed, other transactions are rolled back.

ibase_query (PHP 3>= 3.0.6, PHP 4)

Execute a query on an InterBase database

```
int ibase_query ([int link_identifier, string query [, int bind_args]])
```

Performs a query on an InterBase database, returning a result identifier for use with **ibase_fetch_row()**, **ibase_fetch_object()**, **ibase_free_result()** and **ibase_free_query()**.

Poznámka: Although this function supports variable binding to parameter placeholders, there is not very much meaning using this capability with it. For real life use and an example, see **ibase_prepare()** and **ibase_execute()**.

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4)

Fetch a row from an InterBase database

```
array ibase_fetch_row (int result_identifier)
```

Returns the next row specified by the result identifier obtained using the **ibase_query()**.

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Get an object from a InterBase database

```
object ibase_fetch_object (int result_id)
```

Fetches a row as a pseudo-object from a *result_id* obtained either by **ibase_query()** or **ibase_execute()**.

```
<php
    $dbh = ibase_connect ($host, $username, $password);
```

```
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

See also **ibase_fetch_row()**.

ibase_field_info (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Get information about a field

```
array ibase_field_info (int result, int field number)
```

Returns an array with information about a field after a select query has been run. The array is in the form of name, alias, relation, length, type.

```
// helio@helio.com.br  08-Dec-2000 02:53

$rs=ibase_query("Select * from something");
$coln = ibase_num_fields($rs);
for ($i=0 ; $i < $coln ; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
    echo "relation: ".$col_info['relation']."\n";
    echo "length: ".$col_info['length']."\n";
    echo "type: ".$col_info['type']."\n";
    }
```

ibase_free_result (PHP 3>= 3.0.6, PHP 4)

Free a result set

```
int ibase_free_result (int result_identifier)
```

Free's a result set the has been created by **ibase_query()**.

ibase_prepare (PHP 3>= 3.0.6, PHP 4)

Prepare a query for later binding of parameter placeholders and execution

```
int ibase_prepare ([int link_identifier, string query])
```

Prepare a query for later binding of parameter placeholders and execution (via **ibase_execute()**).

ibase_execute (PHP 3>= 3.0.6, PHP 4)

Execute a previously prepared query

```
int ibase_execute (int query [, int bind_args])
```

Execute a query prepared by **ibase_prepare()**. This is a lot more effective than using **ibase_query()** if you are repeating a same kind of query several times with only some parameters changing.

```
<?php
    $updates = array(
        1 => 'Eric',
        5 => 'Filip',
        7 => 'Larry'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_trans (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Begin a transaction

```
int ibase_trans ([int trans_args [, int link_identifier]])
```

Begins a transaction.

ibase_commit (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Commit a transaction

```
int ibase_commit ([int link_identifier, int trans_number])
```

Commits transaction *trans_number* which was created with **ibase_trans()**.

ibase_rollback (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Rolls back a transaction

```
int ibase_rollback ([int link_identifier, int trans_number])
```

Rolls back transaction *trans_number* which was created with **ibase_trans()**.

ibase_free_query (PHP 3>= 3.0.6, PHP 4)

Free memory allocated by a prepared query

```
int ibase_free_query (int query)
```

Free a query prepared by **ibase_prepare()**.

ibase_timefmt (PHP 3>= 3.0.6, PHP 4)

Sets the format of timestamp, date and time type columns returned from queries

```
int ibase_timefmt (string format [, int columntype])
```

Sets the format of timestamp, date or time type columns returned from queries. Internally, the columns are formatted by c-function strftime(), so refer to it's documentation regarding to the format of the string. *columntype* is one of the constants IBASE_TIMESTAMP, IBASE_DATE and IBASE_TIME. If omitted, defaults to IBASE_TIMESTAMP for backwards compatibility.

```
<?php
    // InterBase 6 TIME-type columns will be returned in
    // the form '05 hours 37 minutes'.
    ibase_timefmt("%H hours %M minutes", IBASE_TIME);
?>
```

You can also set defaults for these formats with PHP configuration directives ibase.timestampformat, ibase.dateformat and ibase.timeformat.

Poznámka: *columntype* was added in PHP 4.0. It has any meaning only with InterBase version 6 and higher.

Poznámka: A backwards incompatible change happened in PHP 4.0 when PHP configuration directive ibase.timeformat was renamed to ibase.timestampformat and directives ibase.dateformat and ibase.timeformat were added, so that the names would match better their functionality.

ibase_num_fields (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Get the number of fields in a result set

```
int ibase_num_fields (int result_id)
```

Returns an integer containing the number of fields in a result set.

```
<?php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);

    if (ibase_num_fields($sth) > 0) {
        while ($row = ibase_fetch_object ($sth)) {
            print $row->email . "\n";
        }
    } else {
        die ("No Results were found for your query");
    }

    ibase_close ($dbh);
?>
```

See also: **ibase_field_info()**.

Poznámka: Ibase_num_fields() is currently not functional in PHP 4.

ibase_errmsg (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Returns error messages

```
string ibase_errmsg (void )
```

Returns a string containing an error message.

XXXVI. Ingres II functions

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the `-with-ingres` option. You need the Open API library and header files included with Ingres II. If the II_SYSTEM environment variable isn't correctly set you may have to use `-with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

> **Poznámka:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

ingres_connect (PHP 4 >= 4.0.2)

Open a connection to an Ingres II database.

```
resource ingres_connect ([string database [, string username [, string password]]])
```

Returns a Ingres II link resource on success, or false on failure.

ingres_connect() opens a connection with the Ingres database designated by *database*, which follows the syntax *[node_id::]dbname[/svr_class]*.

If some parameters are missing, **ingres_connect()** uses the values in `php.ini` for *ingres.default_database*, *ingres.default_user* and *ingres.default_password*.

The connection is closed when the script ends or when **ingres_close()** is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Příklad 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Příklad 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also **ingres_pconnect()**, and **ingres_close()**.

ingres_pconnect (PHP 4 >= 4.0.2)

Open a persistent connection to an Ingres II database.

```
resource ingres_pconnect ([string database [, string username [, string password]]])
```

Returns a Ingres II link resource on success, or false on failure.

See **ingres_connect()** for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and **ingres_connect()** : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ingres_close()** will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also **ingres_connect()**, and **ingres_close()**.

ingres_close (PHP 4 >= 4.0.2)

Close an Ingres II database connection

```
boolingres_close ([resource link])
```

Returns true on success, or false on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also **ingres_connect()**, and **ingres_pconnect()**.

ingres_query (PHP 4 >= 4.0.2)

Send a SQL query to Ingres II

```
boolingres_query (string query [, resource link])
```

Returns true on success, or false on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call **ingres_commit()** to commit the changes made to the database or **ingres_rollback()** to cancel these changes. When the script ends, any open transaction is rolled back (by calling **ingres_rollback()**). You can also use **ingres_autocommit()** before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can't be sent with this function :

- close (see **ingres_close()**).
- commit (see **ingres_commit()**).
- connect (see **ingres_connect()**).
- disconnect (see **ingres_close()**).
- get dbevent
- prepare to commit
- rollback (see **ingres_rollback()**).
- savepoint
- set autocommit (see **ingres_autocommit()**).
- all cursor related queries are unsupported

Příklad 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_fetch_array()**, **ingres_fetch_object()**, **ingres_fetch_row()**, **ingres_commit()**, **ingres_rollback()** and **ingres_autocommit()**.

ingres_num_rows (PHP 4 >= 4.0.2)

Get the number of rows affected or returned by the last query

```
intingres_num_rows ([resource link])
```

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Poznámka:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using **ingres_fetch_array()**, **ingres_fetch_object()** or **ingres_fetch_row()** the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns false, indicating that no more results are available.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_num_fields (PHP 4 >= 4.0.2)

Get the number of fields returned by the last query

```
intingres_num_fields ([resource link])
```

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to **ingres_query()**

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_name (PHP 4 >= 4.0.2)

Get the name of a field in a query result.

```
stringingres_field_name (int index [, resource link])
```

ingres_field_name() returns the name of a field in a query result, or false on failure.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_type (PHP 4 >= 4.0.2)

Get the type of a field in a query result.

```
stringingres_field_type (int index [, resource link])
```

ingres_field_type() returns the type of a field in a query result, or false on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see **ingres_field_length()**). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_nullable (PHP 4 >= 4.0.2)

Test if a field is nullable.

```
boolingres_field_nullable (int index [, resource link])
```

ingres_field_nullable() returns true if the field can be set to the NULL value and false if it can't.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_length (PHP 4 >= 4.0.2)

Get the length of a field.

```
intingres_field_length (int index [, resource link])
```

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_precision (PHP 4 >= 4.0.2)

Get the precision of a field.

```
intingres_field_precision (int index [, resource link])
```

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_scale (PHP 4 >= 4.0.2)

Get the scale of a field.

```
intingres_field_scale (int index [, resource link])
```

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_fetch_array (PHP 4 >= 4.0.2)

Fetch a row of result into an array.

```
arrayingres_fetch_array ([int result_type [, resource link]])
```

ingres_fetch_array() Returns an array that corresponds to the fetched row, or false if there are no more rows.

This function is an extended version of **ingres_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be II_NUM for enumerated array, II_ASSOC for associative array, or II_BOTH (default).

Speed-wise, the function is identical to **ingres_fetch_object()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Příklad 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_fetch_row (PHP 4 >= 4.0.2)

Fetch a row of result into an enumerated array.

```
arrayingres_fetch_row ([resource link])
```

ingres_fetch_row() returns an array that corresponds to the fetched row, or false if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or false if there are no more rows.

Příklad 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_num_fields()**, **ingres_query()**, **ingres_fetch_array()** and **ingres_fetch_object()**.

ingres_fetch_object (PHP 4 >= 4.0.2)

Fetch a row of result into an object.

```
objectingres_fetch_object ([int result_type [, resource link]])
```

ingres_fetch_object() Returns an object that corresponds to the fetched row, or false if there are no more rows.

This function is similar to **ingres_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: II_ASSOC, II_NUM, and II_BOTH.

Speed-wise, the function is identical to **ingres_fetch_array()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Příklad 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_array()** and **ingres_fetch_row()**.

ingres_rollback (PHP 4 >= 4.0.2)

Roll back a transaction.

```
boolingres_rollback ([resource link])
```

ingres_rollback() rolls back the currently open transaction, actualy canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

See also **ingres_query()**, **ingres_commit()** and **ingres_autocommit()**.

ingres_commit (PHP 4 >= 4.0.2)

Commit a transaction.

```
boolingres_commit ([resource link])
```

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

You can also have the server commit automaticaly after every query by calling **ingres_autocommit()** before opening the transaction.

See also **ingres_query()**, **ingres_rollback()** and **ingres_autocommit()**.

ingres_autocommit (PHP 4 >= 4.0.2)

Switch autocommit on or off.

```
boolingres_autocommit ([resource link])
```

ingres_autocommit() is called before opening a transaction (before the first call to **ingres_query()** or just after a call to **ingres_rollback()** or **ingres_autocommit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if **ingres_commit()** was called after every call to **ingres_query()**.

See also **ingres_query()**, **ingres_rollback()** and **ingres_commit()**.

XXXVII. LDAP functions

Introduction to LDAP

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like

```
/usr/local/myapp/docs
```

The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be.

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as ..

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Complete code example

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Příklad 1. LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection

echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is ".$ds."<p>";

if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                           // read-only access
    echo "Bind result is ".$r."<p>";

    echo "Searching for (sn=S*) ...";
    // Search surname entry
    $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
    echo "Search result is ".$sr."<p>";
```

```
    echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";

    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: ". $info[$i]["dn"] ."<br>";
        echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
        echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Closing connection";
    ldap_close($ds);

} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

Using the PHP LDAP calls

You will need to get and compile LDAP client libraries from either the University of Michigan ldap-3.3 package or the Netscape Directory SDK 3.0. You will also need to recompile PHP with LDAP support enabled before PHP's LDAP calls will work.

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My Company,c=US")
- Whether you need a password to access the server (many servers will provide read access for an "anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect()    // establish connection to server
  |
ldap_bind()       // anonymous or authenticated "login"
  |
do something like search or update the directory
and display the results
  |
ldap_close()      // "logout"
```

More Information

Lots of information about LDAP can be found at

- Netscape (http://developer.netscape.com/tech/directory/)
- University of Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- OpenLDAP Project (http://www.openldap.org/)
- LDAP World (http://elvira.innosoft.com/ldapworld)

The Netscape SDK contains a helpful Programmer's Guide in .html format.

ldap_add (PHP 3, PHP 4)

Add entries to LDAP directory

```
int ldap_add (int link_identifier, string dn, array entry)
```

returns true on success and false on error.

The **ldap_add()** function is used to add entries in the LDAP directory. The DN of the entry to be added is specified by dn. Array entry specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

```
entry["attribute1"] = value
entry["attribute2"][0] = value1
entry["attribute2"][1] = value2
```

Příklad 1. Complete example with authenticated bind

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");

    // prepare data
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";

    // add data to directory
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

ldap_bind (PHP 3, PHP 4)

Bind to LDAP directory

```
int ldap_bind (int link_identifier [, string bind_rdn [, string bind_password]])
```

Binds to the LDAP directory with specified RDN and password. Returns true on success and false on error.

ldap_bind() does a bind operation on the directory. bind_rdn and bind_password are optional. If not specified, anonymous bind is attempted.

ldap_close (PHP 3, PHP 4)

Close link to LDAP server

```
int ldap_close (int link_identifier)
```

Returns true on success, false on error.

ldap_close() closes the link to the LDAP server that's associated with the specified *link_identifier*.

This call is internally identical to **ldap_unbind()**. The LDAP API uses the call **ldap_unbind()**, so perhaps you should use this in preference to **ldap_close()**.

ldap_compare (PHP 4 >= 4.0.2)

Compare value of attribute found in entry specified with DN

```
int ldap_compare (int link_identifier, string dn, string attribute, string value)
```

Returns true if *value* matches otherwise returns false. Returns -1 on error.

ldap_compare() is used to compare *value* of *attribute* to value of same attribute in LDAP directory entry specified with *dn*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Příklad 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if(ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Password correct.";
        } elseif ($r === FALSE) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Poznámka: ldap_compare() can NOT be used to compare BINARY values!

Poznámka: This function was added in 4.0.2.

ldap_connect (PHP 3, PHP 4)

Connect to an LDAP server

```
int ldap_connect ([string hostname [, int port]])
```

Returns a positive LDAP link identifier on success, or false on error.

ldap_connect() establishes a connection to a LDAP server on a specified *hostname* and *port*. Both the arguments are optional. If no arguments are specified then the link identifier of the already opened link will be returned. If only *hostname* is specified, then the port defaults to 389.

If you are using OpenLDAP 2.x.x you can specify a URL instead of the hostname. To use LDAP with SSL, compile OpenLDAP 2.x.x with SSL support, configure PHP with SSL, and use ldaps://hostname/ as host parameter. The port parameter is not used when using URLs. URL and SSL support were added in 4.0.4.

ldap_count_entries (PHP 3, PHP 4)

Count the number of entries in a search

```
int ldap_count_entries (int link_identifier, int result_identifier)
```

Returns number of entries in the result or false on error.

ldap_count_entries() returns the number of entries stored in the result of previous search operations. *result_identifier* identifies the internal ldap result.

ldap_delete (PHP 3, PHP 4)

Delete an entry from a directory

```
int ldap_delete (int link_identifier, string dn)
```

Returns true on success and false on error.

ldap_delete() function delete a particular entry in LDAP directory specified by dn.

ldap_dn2ufn (PHP 3, PHP 4)

Convert DN to User Friendly Naming format

```
string ldap_dn2ufn (string dn)
```

ldap_dn2ufn() function is used to turn a DN into a more user-friendly form, stripping off type names.

ldap_err2str (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Convert LDAP error number into string error message

```
string ldap_err2str (int errno)
```

returns string error message.

This function returns the string error message explaining the error number errno. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

See also **ldap_errno()** and **ldap_error()**.

Příklad 1. Enumerating all LDAP error messages

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_err2str($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Return the LDAP error number of the last LDAP command

```
int ldap_errno (int link_id)
```

return the LDAP error number of the last LDAP command for this link.

This function returns the standardized error number returned by the last LDAP command for the given link identifier. This number can be converted into a textual error message using **ldap_err2str()**.

Unless you lower your warning level in your php3.ini sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Příklad 1. Generating and catching an error

```
<?php
// This example contains an error, which we will catch.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// syntax error in filter expression (errno 87),
// must be "objectclass=*" to work.
$res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
    printf("LDAP-Errno: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Error: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d matching entries.<br>\n", $info["count"]);
?>
```

see also **ldap_err2str()** and **ldap_error()**.

ldap_error (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Return the LDAP error message of the last LDAP command

```
string ldap_error (int link_id)
```

returns string error message.

This function returns the string error message explaining the error generated by the last LDAP command for the given link identifier. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Unless you lower your warning level in your `php3.ini` sufficiently or prefix your LDAP commands with `@` (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

see also **ldap_err2str()** and **ldap_errno()**.

ldap_explode_dn (PHP 3, PHP 4)

Splits DN into its component parts

```
array ldap_explode_dn (string dn, int with_attrib)
```

ldap_explode_dn() function is used to split the a DN returned by **ldap_get_dn()** and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN. **ldap_explode_dn()** returns an array of all those components. *with_attrib* is used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *with_attrib* to 0 and to get only values set it to 1.

ldap_first_attribute (PHP 3, PHP 4)

Return first attribute

```
string ldap_first_attribute (int link_identifier, int result_entry_identifier, int
ber_identifier)
```

Returns the first attribute in the entry on success and false on error.

Similar to reading entries, attributes are also read one by one from a particular entry. **ldap_first_attribute()** returns the first attribute in the entry pointed by the entry identifier. Remaining attributes are retrieved by calling **ldap_next_attribute()** successively. *ber_identifier* is the identifier to internal memory location pointer. It is passed by reference. The same *ber_identifier* is passed to the **ldap_next_attribute()** function, which modifies that pointer.

see also **ldap_get_attributes()**

ldap_first_entry (PHP 3, PHP 4)

Return first result id

```
int ldap_first_entry (int link_identifier, int result_identifier)
```

Returns the result entry identifier for the first entry on success and false on error.

Entries in the LDAP result are read sequentially using the **ldap_first_entry()** and **ldap_next_entry()** functions. **ldap_first_entry()** returns the entry identifier for first entry in the result. This entry identifier is then supplied to **lap_next_entry()** routine to get successive entries from the result.

see also **ldap_get_entries()**.

ldap_free_result (PHP 3, PHP 4)

Free result memory

```
int ldap_free_result (int result_identifier)
```

Returns true on success and false on error.

ldap_free_result() frees up the memory allocated internally to store the result and pointed by the *result_identifier*. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the ldap result gets freed at the end of the script. In case the script is making successive searches which return large result sets, **ldap_free_result()** could be called to keep the runtime memory usage by the script low.

ldap_get_attributes (PHP 3, PHP 4)

Get attributes from a search result entry

```
array ldap_get_attributes (int link_identifier, int result_entry_identifier)
```

Returns a complete entry information in a multi-dimensional array on success and false on error.

ldap_get_attributes() function is used to simplify reading the attributes and values from an entry in the search result. The return value is a multi-dimensional array of attributes and values.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won't care what other data is held.

return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute
return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = ith value of the attribute

Příklad 1. Show the list of attributes held for a particular directory entry

```
// $ds is the link identifier for the directory

// $sr is a valid search result from a prior call to
// one of the ldap directory search calls

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"]." attributes held for this entry:<p>";

for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

see also **ldap_first_attribute()** and **ldap_next_attribute()**

ldap_get_dn (PHP 3, PHP 4)

Get the DN of a result entry

```
string ldap_get_dn (int link_identifier, int result_entry_identifier)
```

Returns the DN of the result entry and false on error.

ldap_get_dn() function is used to find out the DN of an entry in the result.

ldap_get_entries (PHP 3, PHP 4)

Get all result entries

```
array ldap_get_entries (int link_identifier, int result_identifier)
```

Returns a complete result information in a multi-dimenasional array on success and false on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result and then reading the attributes and multiple values. The entire information is returned by one function call in a multi-dimensional array. The structure of the array is as follows.

The attribute index is converted to lowercase. (Attributes are case-insensitive for directory servers, but not when used as array indices)

return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] = DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for
attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry

see also **ldap_first_entry()** and **ldap_next_entry()**

ldap_get_option (PHP 4 >= 4.0.4)

Get the current value for given option

```
boolean ldap_get_option (int link_identifier, int option, mixed retval)
```

Sets *retval* to the value of the specified option, and returns true on success and false on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Příklad 1. Check protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Using protocol version $version";
else
```

```
    echo "Unable to determine protocol version";
```

See also **ldap_set_option()**.

ldap_get_values (PHP 3, PHP 4)

Get all values from a result entry

```
array ldap_get_values (int link_identifier, int result_entry_identifier, string
attribute)
```

Returns an array of values for the attribute on success and false on error.

ldap_get_values() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This call needs a *result_entry_identifier*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the **ldap_get_attributes()** call to work out what attributes exist for a given entry.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person’s directory entry all labeled with the attribute "mail"

return_value["count"] = number of values for attribute
return_value[0] = first value of attribute
return_value[i] = ith value of attribute

Příklad 1. List all values of the "mail" attribute for a directory entry

```
// $ds is a valid link identifier for a directory server

// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls

// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry

$values = ldap_get_values($ds, $entry,"mail");

echo $values["count"]." email addresses for this entry.<p>";

for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Get all binary values from a result entry

```
array ldap_get_values_len (int link_identifier, int result_entry_identifier, string
attribute)
```

Returns an array of values for the attribute on success and false on error.

ldap_get_values_len() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *`result_entry_identifier`*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This function is used exactly like **ldap_get_values()** except that it handles binary data and not string data.

> **Poznámka:** This function was added in 4.0.

ldap_list (PHP 3, PHP 4)

Single-level search

```
int ldap_list (int link_identifier, string base_dn, string filter [, array attributes
[, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

Returns a search result identifier or false on error.

ldap_list() performs the search for a specified filter on the directory with the scope LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the base dn given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

This call takes 5 optional parameters. See **ldap_search()** notes.

> **Poznámka:** These optional parameters were added in 4.0.2: *`attrsonly`*, *`sizelimit`*, *`timelimit`*, *`deref`*.

Příklad 1. Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

ldap_modify (PHP 3, PHP 4)

Modify an LDAP entry

```
int ldap_modify (int link_identifier, string dn, array entry)
```

Returns true on success and false on error.

ldap_modify() function is used to modify the existing entries in the LDAP directory. The structure of the entry is same as in **ldap_add()**.

ldap_mod_add (PHP 3>= 3.0.8, PHP 4)

Add attribute values to current attributes

```
int ldap_mod_add (int link_identifier, string dn, array entry)
```

returns true on success and false on error.

This function adds attribute(s) to the specified dn. It performs the modification at the attribute level as opposed to the object level. Object-level additions are done by the **ldap_add()** function.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4)

Delete attribute values from current attributes

```
int ldap_mod_del (int link_identifier, string dn, array entry)
```

returns true on success and false on error.

This function removes attribute(s) from the specified dn. It performs the modification at the attribute level as opposed to the object level. Object-level deletions are done by the **ldap_del()** function.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4)

Replace attribute values with new ones

```
int ldap_mod_replace (int link_identifier, string dn, array entry)
```

returns true on success and false on error.

This function replaces attribute(s) from the specified dn. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the **ldap_modify()** function.

ldap_next_attribute (PHP 3, PHP 4)

Get the next attribute in result

```
string ldap_next_attribute (int link_identifier, int result_entry_identifier, int
ber_identifier)
```

Returns the next attribute in an entry on success and false on error.

ldap_next_attribute() is called to retrieve the attributes in an entry. The internal state of the pointer is maintained by the *ber_identifier*. It is passed by reference to the function. The first call to **ldap_next_attribute()** is made with the *result_entry_identifier* returned from **ldap_first_attribute()**.

see also **ldap_get_attributes()**

ldap_next_entry (PHP 3, PHP 4)

Get next result entry

```
int ldap_next_entry (int link_identifier, int result_entry_identifier)
```

Returns entry identifier for the next entry in the result whose entries are being read starting with **ldap_first_entry()**. If there are no more entries in the result then it returns false.

ldap_next_entry() function is used to retrieve the entries stored in the result. Successive calls to the **ldap_next_entry()** return entries one by one till there are no more entries. The first call to **ldap_next_entry()** is made after the call to **ldap_first_entry()** with the result_identifier as returned from the **ldap_first_entry()**.

see also **ldap_get_entries()**

ldap_read (PHP 3, PHP 4)

Read an entry

```
int ldap_read (int link_identifier, string base_dn, string filter [, array attributes
[, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

Returns a search result identifier or false on error.

ldap_read() performs the search for a specified filter on the directory with the scope LDAP_SCOPE_BASE. So it is equivalent to reading an entry from the directory.

An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of "objectClass=*". If you know which entry types are used on the directory server, you might use an appropriate filter such as "objectClass=inetOrgPerson".

This call takes 5 optional parameters. See **ldap_search()** notes.

> **Poznámka:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

ldap_search (PHP 3, PHP 4)

Search LDAP tree

```
int ldap_search (int link_identifier, string base_dn, string filter [, array attributes
[, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

Returns a search result identifier or false on error.

ldap_search() performs the search for a specified filter on the directory with the scope of LDAP_SCOPE_SUBTREE. This is equivalent to searching the entire directory. *base_dn* specifies the base DN for the directory.

There is a optional fourth parameter, that can be added to restrict the attributes and values returned by the server to just those required. This is much more efficient than the default action (which is to return all attributes and their associated values). The use of the fourth parameter should therefore be considered good practice.

The fourth parameter is a standard PHP string array of the required attributes, eg array("mail","sn","cn") Note that the "dn" is always returned irrespective of which attributes types are requested.

Note too that some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This occurs also if the sixth parameter *sizelimit* has been used to limit the count of fetched entries.

The fifth parameter *attrsonly* should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

With the sixth parameter *sizelimit* it is possible to limit the count of entries fetched. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset sizelimit. You can set it lower though.

The seventh parameter *timelimit* sets the number of seconds how long is spend on the search. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset timelimit. You can set it lower though.

The eigth parameter *deref* specifies how aliases should be handled during the search. It can be one of the following:

- LDAP_DEREF_NEVER - (default) aliases are never dereferenced.
- LDAP_DEREF_SEARCHING - aliases should be dereferenced during the search but not when locating the base object of the search.
- LDAP_DEREF_FINDING - aliases should be dereferenced when locating the base object but not during the search.
- LDAP_DEREF_ALWAYS - aliases should be dereferenced always.

These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

The search filter can be simple or advanced, using boolean operators in the format described in the LDAP doumentation (see the Netscape Directory SDK (http://developer.netscape.com/docs/manuals/directory/41/ag/find.htm) for full information on filters).

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.

Příklad 1. LDAP search

```
// $ds is a valid link identifier for a directory server

// $person is all or part of a person’s name, eg "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entries returned<p>";
```

ldap_set_option (PHP 4 >= 4.0.4)

Set the value of the given option

```
boolean ldap_set_option (int link_identifier, int option, mixed newval)
```

Sets the value of the specified option to be *newval*, and returns true on success and false on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here’s a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt) for details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The

optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to *FALSE* if not supplied. See also the second example below.

This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Příklad 1. Set protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Using LDAPv3";
else
    echo "Failed to set protocol version to 3";
```

Příklad 2. Set server controls

```
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
```

See also **ldap_get_option()**.

ldap_unbind (PHP 3, PHP 4)

Unbind from LDAP directory

```
int ldap_unbind (int link_identifier)
```

Returns true on success and false on error.

ldap_unbind() function unbinds from the LDAP directory.

XXXVIII. Mailové funkce

Funkce **mail()** umožňuje odesílat maily.

mail (PHP 3, PHP 4)

send mail

```
bool mail (string to, string subject, string message [, string additional_headers])
```

Mail() automaticky odmailuje vzkaz specifikovaný v *message* příjemci specifikovanému v *to*. Přidáním čárky mezi adresami v *to* můžete specifikovat více příjemců.

Příklad 1. Odeslání mailu.

```
mail("rasmus@lerdorf.on.ca", "Můj předmět", "Řádek 1\nŘádek 2\nŘádek 3");
```

Pokud je předán čtvrtý argument, jeho hodnota se vloží na konec hlaviček. Toto se obvykle používá k přidání extra hlaviček. Vícenásobné hlavičky se oddělují zařádkováním.

Příklad 2. Odeslání mailu s extra hlavičkami.

```
mail("nobody@aol.com", "předmět", $message,
     "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-
Mailer: PHP/" . phpversion());
```

K vytvoření komplexních emailů můžete také použít poměrně jednoduché techniky pro tvorbu řetězců.

Příklad 3. Odeslání komplexního emailu

```
/* recipients */
$recipient .= "Mary <mary@u.college.edu>" . ", " ; //všimněte si čárky
$recipient .= "Kelly <kelly@u.college.edu> . ", ";
$recipient .= "ronabop@php.net";

/* subject */
$subject = "Birthday Reminders for August";

/* message */
$message .= "Následující email obsahuje formátovanou ASCII tabulku\n";
$message .= "Day \t\tMonth \t\tYear\n";
$message .= "3rd \t\tAug \t\t1970\n";
$message .= "17rd\t\tAug \t\t1973\n";

/* můžete přidat signaturu */
$message .= "-\r\n"; //oddělovač signatury
$message .= "Birthday reminder copylefted by public domain";

/* dodatečné hlavičky pro chyby, From, cc, bcc, atd */

$headers .= "From: Birthday Reminder <birthday@php.net>\n";
$headers .= "X-Sender: <birthday@php.net>\n";
$headers .= "X-Mailer: PHP\n"; // mailový klient
$headers .= "X-Priority: 1\n"; // Urgentní vzkaz!
$headers .= "Return-Path: <birthday@php.net>\n";  // Návratová cesta pro chyby

/* Pokud chcete poslat HTML email, odkomentujte následující řádek */
// $headers .= "Content-Type: text/html; charset=iso-8859-1\n"; // Mime typ

$headers .= "cc:birthdayarchive@php.net\n"; // CC
$headers .= "bcc:birthdaycheck@php.net, birthdaygifts@php.net\n"; // BCC

/* a ted' to odešleme */
mail($recipient, $subject, $message, $headers);
```

ezmlm_hash (PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

Počítá hash hodnotu potřebnou pro EZMLM

```
int ezmlm_hash (string addr)
```

ezmlm_hash() Počítá hash hodnotu, která je potřeba pro uchovávání EZMLM mailing listů v MySQL databázi.

Příklad 1. Výpočet hashe a přihlášení uživatele

```
$user = "kris@koehntopp.de";
$hash = ezmlm_hash ($user);
$query = sprintf ("INSERT INTO sample VALUES (%s, '%s')", $hash, $user);
$db->query($query); // použito databázové rozhraní PHPLIB
```

XXXIX. Mathematical Functions

Introduction

These math functions will only handle values within the range of the long and double types on your computer. If you need to handle bigger numbers, take a look at the arbitrary precision math functions.

Math constants

The following values are defined as constants in PHP by the math extension:

Tabulka 1. Math constants

Constant	Value	Description
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Euler constant [4.0.2]

Only M_PI is available in PHP versions up to and including PHP4RC1. All other constants are available starting with PHP 4.0. Constants labelled [4.0.2] were added in PHP 4.0.2.

abs (PHP 3, PHP 4)

Absolute value

```
mixed abs (mixed number)
```

Returns the absolute value of number. If the argument number is float, return type is also float, otherwise it is int.

acos (PHP 3, PHP 4)

Arc cosine

```
float acos (float arg)
```

Returns the arc cosine of arg in radians.

See also **asin()** and **atan()**.

asin (PHP 3, PHP 4)

Arc sine

```
float asin (float arg)
```

Returns the arc sine of arg in radians.

See also **acos()** and **atan()**.

atan (PHP 3, PHP 4)

Arc tangent

```
float atan (float arg)
```

Returns the arc tangent of arg in radians.

See also **asin()** and **acos()**.

atan2 (PHP 3>= 3.0.5, PHP 4)

arc tangent of two variables

```
float atan2 (float y, float x)
```

This function calculates the arc tangent of the two variables *x* and *y*. It is similar to calculating the arc tangent of *y* / *x*, except that the signs of both arguments are used to determine the quadrant of the result.

The function returns the result in radians, which is between -PI and PI (inclusive).

See also **acos()** and **atan()**.

base_convert (PHP 3>= 3.0.6, PHP 4)

Convert a number between arbitrary bases

```
string base_convert (string number, int frombase, int tobase)
```

Returns a string containing *number* represented in base *tobase*. The base in which *number* is given is specified in *frombase*. Both *frombase* and *tobase* have to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, with a meaning 10, b meaning 11 and z meaning 35.

Příklad 1. Base_convert()

```
$binary = base_convert ($hexadecimal, 16, 2);
```

bindec (PHP 3, PHP 4)

Binary to decimal

```
int bindec (string binary_string)
```

Returns the decimal equivalent of the binary number represented by the binary_string argument.

Octdec converts a binary number to a decimal number. The largest number that can be converted is 31 bits of 1's or 2147483647 in decimal.

See also the **decbin()** function.

ceil (PHP 3, PHP 4)

Round fractions up

```
int ceil (float number)
```

Returns the next highest integer value from *number*. Using **ceil()** on integers is absolutely a waste of time.

```
$x = ceil(4.25);
//  which would make $x=5
```

NOTE: PHP/FI 2's **ceil()** returned a float. Use: `$new = (double)ceil($number);` to get the old behaviour.

See also **floor()** and **round()**.

cos (PHP 3, PHP 4)

Cosine

```
float cos (float arg)
```

Returns the cosine of arg in radians.

See also **sin()** and **tan()**.

decbin (PHP 3, PHP 4)

Decimal to binary

```
string decbin (int number)
```

Returns a string containing a binary representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to a string of 31 1's.

See also the **bindec()** function.

dechex (PHP 3, PHP 4)

Decimal to hexadecimal

```
string dechex (int number)
```

Returns a string containing a hexadecimal representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to "7fffffff".

See also the **hexdec()** function.

decoct (PHP 3, PHP 4)

Decimal to octal

```
string decoct (int number)
```

Returns a string containing an octal representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to "17777777777".

See also **octdec()**.

deg2rad (PHP 3>= 3.0.4, PHP 4)

Converts the number in degrees to the radian equivalent

```
double deg2rad (double number)
```

This function converts *number* from degrees to the radian equivalent.

See also **rad2deg()**.

exp (PHP 3, PHP 4)

e to the power of ...

```
float exp (float arg)
```

Returns e raised to the power of *arg*.

See also **pow()**.

floor (PHP 3, PHP 4)

Round fractions down

```
int floor (float number)
```

Returns the next lowest integer value from *number*. Using **floor()** on integers is absolutely a waste of time.

NOTE: PHP/FI 2's **floor()** returned a float. Use: `$new = (double)floor($number);` to get the old behaviour.

See also **ceil()** and **round()**.

getrandmax (PHP 3, PHP 4)

Show largest possible random value

```
int getrandmax (void)
```

Returns the maximum value that can be returned by a call to **rand()**.

See also **rand()**, **srand()**, **mt_rand()**, **mt_srand()**, and **mt_getrandmax()**.

hexdec (PHP 3, PHP 4)

Hexadecimal to decimal

```
int hexdec (string hex_string)
```

Returns the decimal equivalent of the hexadecimal number represented by the hex_string argument. HexDec converts a hexadecimal string to a decimal number. The largest number that can be converted is 7fffffff or 2147483647 in decimal.

See also the **dechex()** function.

lcg_value (PHP 4 >= 4.0b4)

Combined linear congruential generator

```
double lcg_value(void);
```

lcg_value() returns a pseudo random number in the range of (0, 1). The function combines two CGs with periods of 2^31 - 85 and 2^31 - 249. The period of this function is equal to the product of both primes.

log (PHP 3, PHP 4)

Natural logarithm

```
float log (float arg)
```

Returns the natural logarithm of arg.

log10 (PHP 3, PHP 4)

Base-10 logarithm

```
float log10 (float arg)
```

Returns the base-10 logarithm of *arg*.

max (PHP 3, PHP 4)

Find highest value

```
mixed max (mixed arg1, mixed arg2, mixed argn)
```

max() returns the numerically highest of the parameter values.

If the first parameter is an array, **max()** returns the highest value in that array. If the first parameter is an integer, string or double, you need at least two parameters and **max()** returns the biggest of these values. You can compare an unlimited number of values.

If one or more of the values is a double, all the values will be treated as doubles, and a double is returned. If none of the values is a double, all of them will be treated as integers, and an integer is returned.

min (PHP 3, PHP 4)

Find lowest value

```
mixed min (mixed arg1, mixed arg2, mixed argn)
```

Min() returns the numerically lowest of the parameter values.

If the first parameter is an array, **min()** returns the lowest value in that array. If the first parameter is an integer, string or double, you need at least two parameters and **min()** returns the lowest of these values. You can compare an unlimited number of values.

If one or more of the values is a double, all the values will be treated as doubles, and a double is returned. If none of the values is a double, all of them will be treated as integers, and an integer is returned.

mt_rand (PHP 3>= 3.0.6, PHP 4)

Generate a better random value

```
int mt_rand ([int min [, int max]])
```

Many random number generators of older libcs have dubious or unknown characteristics and are slow. By default, PHP uses the libc random number generator with the **rand()** function. **mt_rand()** function is a drop-in replacement for this. It uses a random number generator with known characteristics, the Mersenne Twister, which will produce random numbers that should be suitable for seeding some kinds of cryptography (see the home pages for details) and is four times faster than what the average libc provides. The Homepage of the Mersenne Twister can be found at http://www.math.keio.ac.jp/~matumoto/emt.html, and an optimized version of the MT source is available from http://www.scp.syr.edu/~marc/hawk/twister.html (http://www.scp.syr.edu/~marc/hawk/twister.html).

If called without the optional *min*, *max* arguments **mt_rand()** returns a pseudo-random value between 0 and RAND_MAX. If you want a random number between 5 and 15 (inclusive), for example, use `mt_rand (5, 15)`.

Remember to seed the random number generator before use with **mt_srand()**.

Poznámka: In versions before 3.0.7 the meaning of *max* was *range*. To get the same results in these versions the short example should be `mt_rand (5, 11)` to get a random number between 5 and 15.

See also **mt_srand()**, **mt_getrandmax()**, **srand()**, **rand()** and **getrandmax()**.

mt_srand (PHP 3>= 3.0.6, PHP 4)

Seed the better random number generator

```
void mt_srand (int seed)
```

Seeds the random number generator with *seed*.

```
// seed with microseconds since last "whole" second
mt_srand ((double) microtime() * 1000000);
$randval = mt_rand();
```

See also **mt_rand()**, **mt_getrandmax()**, **srand()**, **rand()**, and **getrandmax()**.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4)

Show largest possible random value

```
int mt_getrandmax (void)
```

Returns the maximum value that can be returned by a call to **mt_rand()**.

See also **mt_rand()**, **mt_srand() rand()**, **srand()**, and **getrandmax()**.

number_format (PHP 3, PHP 4)

Format a number with grouped thousands

```
string number_format (float number, int decimals, string dec_point, string
thousands_sep)
```

Number_format() returns a formatted version of *number*. This function accepts either one, two or four parameters (not three):

If only one parameter is given, *Number* will be formatted without decimals, but with a comma (",") between every group of thousands.

If two parameters are given, *number* will be formatted with *decimals* decimals with a dot (".") in front, and a comma (",") between every group of thousands.

If all four parameters are given, *number* will be formatted with *decimals* decimals, *dec_point* instead of a dot (".") before the decimals and *thousands_sep* instead of a comma (",") between every group of thousands.

octdec (PHP 3, PHP 4)

Octal to decimal

```
int octdec (string octal_string)
```

Returns the decimal equivalent of the octal number represented by the octal_string argument. OctDec converts an octal string to a decimal number. The largest number that can be converted is 17777777777 or 2147483647 in decimal.

See also **decoct()**.

pi (PHP 3, PHP 4)

Get value of pi

```
double pi (void)
```

Returns an approximation of pi.

pow (PHP 3, PHP 4)

Exponential expression

```
float pow (float base, float exp)
```

Returns base raised to the power of exp.

See also **exp()**.

rad2deg (PHP 3>= 3.0.4, PHP 4)

Converts the radian number to the equivalent number in degrees

```
double rad2deg (double number)
```

This function converts *number* from radian to degrees.

See also **deg2rad()**.

rand (PHP 3, PHP 4)

Generate a random value

```
int rand ([int min [, int max]])
```

If called without the optional *min*, *max* arguments **rand()** returns a pseudo-random value between 0 and RAND_MAX. If you want a random number between 5 and 15 (inclusive), for example, use `rand (5, 15)`.

Remember to seed the random number generator before use with **srand()**.

> **Poznámka:** In versions before 3.0.7 the meaning of *max* was *range*. To get the same results in these versions the short example should be `rand (5, 11)` to get a random number between 5 and 15.

See also **srand()**, **getrandmax()**, **mt_rand()**, **mt_srand()**, and **mt_getrandmax()**.

round (PHP 3, PHP 4)

Rounds a float

```
double round (double val [, int precision])
```

Returns the rounded value of *val* to specified *precision* (number of digits after the decimal point).

```
$foo = round (3.4);  // $foo == 3.0
$foo = round (3.5);  // $foo == 4.0
$foo = round (3.6);  // $foo == 4.0

$foo = round (1.95583, 2);  // $foo == 1.96
```

Poznámka: The *precision* parameter is only available in PHP 4.

See also **ceil()** and **floor()**.

sin (PHP 3, PHP 4)

Sine

```
float sin (float arg)
```

Returns the sine of arg in radians.

See also **cos()** and **tan()**.

sqrt (PHP 3, PHP 4)

Square root

```
float sqrt (float arg)
```

Returns the square root of *arg*.

srand (PHP 3, PHP 4)

Seed the random number generator

```
void srand (int seed)
```

Seeds the random number generator with *seed*.

```
// seed with microseconds since last "whole" second
srand ((double) microtime() * 1000000);
$randval = rand();
```

See also **rand()**, **getrandmax()**, **mt_rand()**, **mt_srand()**, and **mt_getrandmax()**.

tan (PHP 3, PHP 4)

Tangent

```
float tan (float arg)
```

Returns the tangent of arg in radians.

See also **sin()** and **cos()**.

XL. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It’s written to be very modular, with plugable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and reoccuring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `-with-mcal`. That requires the mcal library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUGT, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

mcal_open (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Opens up an MCAL connection

```
int mcal_open (string calendar, string username, string password, int options)
```

Returns an MCAL stream on success, false on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Opens up a persistant MCAL connection

```
int mcal_popen (string calendar, string username, string password, int options)
```

Returns an MCAL stream on success, false on error.

mcal_popen() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_reopen (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Reopens an MCAL connection

```
int mcal_reopen (string calendar, int options)
```

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_close (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Close an MCAL stream

```
int mcal_close (int mcal_stream, int flags)
```

Closes the given mcal stream.

mcal_create_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Create a new MCAL calendar

```
string mcal_create_calendar (int stream, string calendar)
```

Creates a new calendar named *calendar*.

mcal_rename_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Rename an MCAL calendar

```
string mcal_rename_calendar (int stream, string old_name, string new_name)
```

Renames the calendar *old_name* to *new_name*.

mcal_delete_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Delete an MCAL calendar

```
string mcal_delete_calendar (int stream, string calendar)
```

Deletes the calendar named *calendar*.

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fetches an event from the calendar stream

```
object mcal_fetch_event (int mcal_stream, int event_id [, int options])
```

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.
Returns an event object consisting of:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

The possible values for recur_type are:

- 0 - Indicates that this event does not recur
- 1 - This event recurs daily
- 2 - This event recurs on a weekly basis
- 3 - This event recurs monthly on a specific day of the month (e.g. the 10th of the month)
- 4 - This event recurs monthly on a sequenced day of the week (e.g. the 3rd Saturday)
- 5 - This event recurs on an annual basis

mcal_list_events (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return a list of IDs for a date or a range of dates.

```
array mcal_list_events (int mcal_stream, objectbegin_date [, object end_date])
```

Returns an array of ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an beginning date and an optional end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_append_event (PHP 4 >= 4.0RC1)

Store a new event into an MCAL calendar

```
int mcal_append_event (int mcal_stream)
```

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the id of the newly inserted event.

mcal_store_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Modify an existing event in an MCAL calendar

```
int mcal_store_event (int mcal_stream)
```

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns the event id of the modified event on success and false on error.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Delete an event from an MCAL calendar

```
int mcal_delete_event (int mcal_stream [, int event_id])
```

mcal_delete_event() deletes the calendar event specified by the event_id.

Returns true.

mcal_snooze (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Turn off an alarm for an event

```
int mcal_snooze (int id)
```

mcal_snooze() turns off an alarm for a calendar event specified by the id.

Returns true.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return a list of events that has an alarm triggered at the given datetime

```
array mcal_list_alarms (int mcal_stream [, int begin_year [, int begin_month [, int
begin_day [, int end_year [, int end_month [, int end_day]]]]]])
```

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_event_init (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Initializes a streams global event structure

```
int mcal_event_init (int stream)
```

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns true.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the category of the streams global event structure

```
int mcal_event_set_category (int stream, string category)
```

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns true.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the title of the streams global event structure

```
int mcal_event_set_title (int stream, string title)
```

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns true.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the description of the streams global event structure

```
int mcal_event_set_description  (int stream, string description)
```

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns true.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the start date and time of the streams global event structure

```
int mcal_event_set_start (int stream, int year, int month [, int day [, int hour [, int
min [, int sec]]]])
```

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns true.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the end date and time of the streams global event structure

```
int mcal_event_set_end (int stream, int year, int month [, int day [, int hour [, int
min [, int sec]]]])
```

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns true.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the alarm of the streams global event structure

```
int mcal_event_set_alarm (int stream, int alarm)
```

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns true.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the class of the streams global event structure

```
int mcal_event_set_class (int stream, int class)
```

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns true.

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns if the given year is a leap year or not

```
int mcal_is_leap_year (int year)
```

mcal_is_leap_year() returns 1 if the given year is a leap year, 0 if not.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the number of days in the given month

```
int mcal_days_in_month (int month, int leap year)
```

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns true if the given year, month, day is a valid date

```
int mcal_date_valid (int year, int month, int day)
```

mcal_date_valid() Returns true if the given year, month and day is a valid date, false if not.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns true if the given year, month, day is a valid time

```
int mcal_time_valid (int hour, int minutes, int seconds)
```

mcal_time_valid() Returns true if the given hour, minutes and seconds is a valid time, false if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the day of the week of the given date

```
int mcal_day_of_week (int year, int month, int day)
```

mcal_day_of_week() returns the day of the week of the given date. Possible return values range from 0 for Sunday through 6 for Saturday.

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the day of the year of the given date

```
int mcal_ (int year, int month, int day)
```

mcal_day_of_year() returns the day of the year of the given date.

mcal_date_compare (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Compares two dates

```
int mcal_date_compare (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)
```

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively.

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the next recurrence of the event

```
int mcal_next_recurrence (int stream, int weekstart, array next)
```

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_none  (int stream)
```

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to MCAL_RECUR_NONE).

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_daily  (int stream, int year, int month, int day, int interval)
```

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_weekly  (int stream, int year, int month, int day, int
interval, int weekdays)
```

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_monthly_mday  (int stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_monthly_wday  (int stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_yearly  (int stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date.

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns an object containing the current streams event structure

```
object mcal_fetch_current_stream_event  (int stream)
```

mcal_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.

- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Adds an attribute and a value to the streams global event structure

```
void mcal_event_add_attribute  (int stream, string attribute, string value)
```

mcal_event_add_attribute() adds an attribute to the stream’s global event structure with the value given by "value".

mcal_expunge (unknown)

Deletes all events marked for being expunged.

```
int mcal_expunge  (int stream)
```

mcal_expunge() Deletes all events which have been previously marked for deletion.

XLI. Mcrypt Encryption Functions

These functions work using mcrypt (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt).

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

If you linked against libmcrypt 2.4.x, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x another cipher mode is also available; nOFB.

To use it, download libmcrypt-x.x.tar.gz from here (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt) and follow the included installation instructions. You need to compile PHP with the `-with-mcrypt` parameter to enable this extension. Make sure you compile libmcrypt with the option `-disable-posix-threads`.

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (**mcrypt_cfb()**, **mcrypt_cbc()**, **mcrypt_ecb()**, and **mcrypt_ofb()**) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Příklad 1. Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

If you linked against libmcrypt 2.4.x, these functions are still available, but it is recommended that you use the advanced functions.

Příklad 2. Encrypt an input value with TripleDES under 2.4.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$td = mcrypt_module_open (MCRYPT_TripleDES, "", MCRYPT_MODE_ECB, "");
$iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
mcrypt_generic_init ($td, $key, $iv);
$encrypted_data = mcrypt_generic ($td, $input);
mcrypt_generic_end ($td);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x mcrypt can also operate in the block cipher mode nOFB and in STREAM mode. Then there are also constants in the form MCRYPT_MODE_mode for use with several functions. We will outline the normal use for each of these modes. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It’s insecure (because it operates in 8bit mode) so it is not recommended to use it.
- nOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the

algorithm.

- STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

PHP does not support encrypting/decrypting bit streams currently. As of now, PHP only supports handling of strings.

For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the mcrypt-2.4.x API these constants also work, but it is possible to specify the name of the cipher as a string with a call to **mcrypt_module_open()**.

Here is a short list of ciphers which are currently supported by the mcrypt extension. If a cipher is not listed here, but is

listed by mcrypt as supported, you can safely assume that this documentation is outdated.

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt 2.4.x only)
- MCRYPT_MARS (libmcrypt 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_192 (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt 2.4.x only)
- MCRYPT_SERPENT (libmcrypt 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x versions)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt 2.4.x only)
- MCRYPT_XTEA (libmcrypt 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher

function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4)

Get the name of the specified cipher

```
string mcrypt_get_cipher_name (int cipher)
```

```
string mcrypt_get_cipher_name (string cipher)
```

Mcrypt_get_cipher_name() is used to get the name of the specified cipher.

Mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x) and returns the name of the cipher or false, if the cipher does not exist.

Příklad 1. Mcrypt_get_cipher_name() Example

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
3DES
```

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4)

Get the block size of the specified cipher

```
int mcrypt_get_block_size (int cipher)
int mcrypt_get_block_size (string cipher, string module)
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_get_block_size() is used to get the size of a block of the specified *cipher*.

Mcrypt_get_block_size() takes one or two arguments, the *cipher* and *module*, and returns the size in bytes.

See also: **mcrypt_get_key_size()**.

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4)

Get the key size of the specified cipher

```
int mcrypt_get_key_size (int cipher)
int mcrypt_get_key_size (string cipher, string module)
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_get_key_size() is used to get the size of a key of the specified *cipher*.

Mcrypt_get_key_size() takes one or two arguments, the *cipher* and *module*, and returns the size in bytes.

See also: **mcrypt_get_block_size()**.

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4)

Create an initialization vector (IV) from a random source

```
string mcrypt_create_iv (int size, int source)
```

Mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *size* determines the size of the IV, *source* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.

Příklad 1. Mcrypt_create_iv() example

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size ($cipher);
$iv = mcrypt_create_iv ($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in CBC mode

```
string mcrypt_cbc (int cipher, string key, string data, int mode [, string iv])
```

```
string mcrypt_cbc (string cipher, string key, string data, int mode [, string iv])
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_cbc() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CBC cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the optional initialization vector.

See also: **mcrypt_cfb()**, **mcrypt_ecb()**, and **mcrypt_ofb()**.

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in CFB mode

```
string mcrypt_cfb (int cipher, string key, string data, int mode, string iv)
```

```
string mcrypt_cfb (string cipher, string key, string data, int mode [, string iv])
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_cfb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: **mcrypt_cbc()**, **mcrypt_ecb()**, and **mcrypt_ofb()**.

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in ECB mode

```
string mcrypt_ecb (int cipher, string key, string data, int mode)

string mcrypt_ecb (string cipher, string key, string data, int mode [, string iv])
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_ecb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in ECB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

See also: **mcrypt_cbc()**, **mcrypt_cfb()**, and **mcrypt_ofb()**.

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in OFB mode

```
string mcrypt_ofb (int cipher, string key, string data, int mode, string iv)

string mcrypt_ofb (string cipher, string key, string data, int mode [, string iv])
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_ofb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in OFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: **mcrypt_cbc()**, **mcrypt_cfb()**, and **mcrypt_ecb()**.

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Get an array of all supported ciphers

```
array mcrypt_list_algorithms ([string lib_dir])
```

Mcrypt_list_algorithms() is used to get an array of all supported algorithms in the

lib_dir. **Mcrypt_list_algorithms()** takes as optional parameter a directory which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir php.ini directive is used.

Příklad 1. Mcrypt_list_algorithms() Example

```
<?php
$algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");

foreach ($algorithms as $cipher) {
echo $cipher."/n";
}
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes (PHP 4 >= 4.0.2)

Get an array of all supported modes

```
array mcrypt_list_modes ([string lib_dir])
```

Mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

Mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir php.ini directive is used.

Příklad 1. Mcrypt_list_modes() Example

```
<?php
$modes = mcrypt_list_modes ();

foreach ($modes as $mode) {
echo "$mode </br>";
}
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV belonging to a specific cipher/mode combination

```
int mcrypt_get_iv_size (string cipher, string mode)
int mcrypt_get_iv_size (resource td)
```

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

Mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns FALSE. If the IV is ignored in the specified cipher/mode combination zero is returned.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

Td is the algorithm specified.

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypts plaintext with given parameters

```
string mcrypt_encrypt (string cipher, string key, string data, string mode [, string
iv])
```

Mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Příklad 1. Mcrypt_encrypt() Example

```
<?php
$iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYPT_RAND)
$key = "This is a very secret key";
$text = "Meet me at 11 o'clock behind the monument.";
echo strlen ($text)."\n";

$crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

mcrypt_decrypt (PHP 4 >= 4.0.2)

Decrypts crypttext with given parameters

```
string mcrypt_decrypt (string cipher, string key, string data, string mode [, string
iv])
```

Mcrypt_decrypt() decrypts the data and returns the unencrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '\0'.

Data is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

mcrypt_module_open (PHP 4 >= 4.0.2)

This function opens the module of the algorithm and the mode to be used

```
resource mcrypt_module_open (string algorithm, string algorithm_directory, string mode,
string mode_directory)
```

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg "twofish" or is one of the MCRYPT_ciphername constants. The library is closed by calling **mcrypt_module_close()**, but there is no need to call that function if **mcrypt_generic_end()** is called. Normally it returns an encryption descriptor, or FALSE on error.

The *algorithm_directory* and *mode_directory* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *mcrypt.algorithms_dir* or *mcrypt.modes_dir* ini-directive is used. When these are not set, the default directory are used that are compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Příklad 1. Mcrypt_module_open() Example

```
<?php
$td = mcrypt_module_open (MCRYPT_DES, "", MCRYPT_MODE_ECB, "/usr/lib/mcrypt-modes");
?>
```

The above example will try to open the DES cipher from the default directory and the EBC mode from the directory /usr/lib/mcrypt-modes.

mcrypt_generic_init (PHP 4 >= 4.0.2)

This function initializes all buffers needed for encryption

```
int mcrypt_generic_init (resource td, string key, string iv)
```

The maximum length of the key should be the one obtained by calling **mcrypt_enc_get_key_size()** and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling **mcrypt_enc_get_iv_size()**. IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns (-1) on error.

You need to call this function before every **mcrypt_generic()** or **mdecrypt_generic()**.

mcrypt_generic (PHP 4 >= 4.0.2)

This function encrypts data

```
string mcrypt_generic (resource td, string data)
```

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

mdecrypt_generic (PHP 4 >= 4.0.2)

This function decrypts data

```
string mdecrypt_generic (resource td, string data)
```

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Příklad 1. Mdecrypt_generic() Example

```
<?php
$iv_size = mcrypt_enc_get_iv_size ($td));
$iv = @mcrypt_create_iv ($iv_size, MCRYPT_RAND);

if (@mcrypt_generic_init ($td, $key, $iv) != -1)
{
    $c_t = mcrypt_generic ($td, $plain_text);
    @mcrypt_generic_init ($td, $key, $iv);
    $p_t = mdecrypt_generic ($td, $c_t);
}
if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0)
    echo "ok";
else
    echo "error";
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption.

mcrypt_generic_end (PHP 4 >= 4.0.2)

This function terminates encryption

```
bool mcrypt_generic_end (resource td)
```

This function terminates encryption specified by the encryption descriptor (td). Actually it clears all buffers, and closes all the modules used. Returns FALSE on error, or TRUE on succes.

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the opened module

```
int mcrypt_enc_self_test (resource td)
```

This function runs the self test on the algorithm specified by the descriptor td. If the self test succeeds it returns zero. In case of an error, it returns 1.

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Checks whether the encryption of the opened mode works on blocks

```
int mcrypt_enc_is_block_algorithm_mode (resource td)
```

This function returns 1 if the mode is for use with block algorithms, otherwise it returns 0. (eg. 0 for stream, and 1 for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Checks whether the algorithm of the opened mode is a block algorithm

```
int mcrypt_enc_is_block_algorithm (resource td)
```

This function returns 1 if the algorithm is a block algorithm, or 0 if it is a stream algorithm.

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Checks whether the opened mode outputs blocks

```
int mcrypt_enc_is_block_mode (resource td)
```

This function returns 1 if the mode outputs blocks of bytes or 0 if it outputs bytes. (eg. 1 for cbc and ecb, and 0 for cfb and stream).

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the opened algorithm

```
int mcrypt_enc_get_block_size (resource td)
```

This function returns the block size of the algorithm specified by the encryption descriptor td in bytes.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

```
int mcrypt_enc_get_key_size (resource td)
```

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor td in bytes.

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

```
array mcrypt_enc_get_supported_key_sizes (resource td)
```

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and **mcrypt_enc_get_key_size()** are supported by the algorithm.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV of the opened algorithm

```
int mcrypt_enc_get_iv_size (resource td)
```

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns '0' then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Returns the name of the opened algorithm

```
string mcrypt_enc_get_algorithms_name (resource td)
```

This function returns the name of the algorithm.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Returns the name of the opened mode

```
string mcrypt_enc_get_modes_name (resource td)
```

This function returns the name of the mode.

mcrypt_module_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the specified module

```
bool mcrypt_module_self_test (string algorithm [, string lib_dir])
```

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns TRUE if the self test succeeds, or FALSE when if fails.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

This function returns if the the specified module is a block algorithm or not

```
bool mcrypt_module_is_block_algorithm_mode (string mode [, string lib_dir])
```

This function returns TRUE if the mode is for use with block algorithms, otherwise it returns 0. (eg. 0 for stream, and 1 for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

This function checks whether the specified algorithm is a block algorithm

```
bool mcrypt_module_is_block_algorithm (string algorithm [, string lib_dir])
```

This function returns TRUE if the specified algorithm is a block algorithm, or FALSE is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

This function returns if the the specified mode outputs blocks or not

```
bool mcrypt_module_is_block_mode (string mode [, string lib_dir])
```

This function returns TRUE if the mode outputs blocks of bytes or FALSE if it outputs just bytes. (eg. 1 for cbc and ecb, and 0 for cfb and stream). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the specified algorithm

```
int mcrypt_module_get_algo_block_size (string algorithm [, string lib_dir])
```

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

```
int mcrypt_module_get_algo_key_size (string algorithm [, string lib_dir])
```

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_supported_key_sizes (unknown)

Returns an array with the supported keysizes of the opened algorithm

```
array mcrypt_module_enc_get_algo_supported_key_sizes (string algorithm [, string
lib_dir])
```

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and **mcrypt_module_get_algo_key_size()** are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

XLII. Mhash funkce

Tyto funkce jsou určeny pro práci s mhash (http://mhash.sourceforge.net/).

Toto je interface ke knihvně mhash. mhash podporuje širokou škálu hash algoritmů jako např. MD5, SHA1, GOST a mnoho jiných.

Pokud chcete tyto funkce používat, stáhněte si mhash distribuci z its web site (http://mhash.sourceforge.net/) a postupujte podle přiložených instrukcí k instalaci. K aktivaci tohoto modulu budete muset zkompilovat PHP s volbou `-with-mhash`

Mhash se dá použít k vytváření kontrolních součtů, message digests, message authentication codes, and more.

Příklad 1. Compute the MD5 digest and hmac and print it out as hex

```
<?php
$input = "what do ya want for nothing?";
$hash = mhash (MHASH_MD5, $input);
print "The hash is ".bin2hex ($hash)."\n<br>";
$hash = mhash (MHASH_MD5, $input, "Jefe");
print "The hmac is ".bin2hex ($hash)."\n<br>";
?>
```

This will produce:

```
The hash is d03cb659cbf9192dcd066272249f8412
The hmac is 750c783e6ab0b503eaa86e310a5db738
```

Kompletní seznam podporovaných hashů viz dokumentaci mhash. Obecným pravidlem je, že hash algoritmus je dostupný z PHP pomocí MHASH_NAZEVHASHE. Například TIGER se v PHP používá pomocí konstanty MHASH_TIGER.

Zde je seznam hashů podporovaných mhashem v současné době. Pokud zde není některý hash jmenován, ale v dokumentaci mhashe je uveden jako podporovaný, můžete bezpečně předpokládat, že je tato dokumentace zastaralá.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL256
- MHASH_HAVAL192
- MHASH_HAVAL160
- MHASH_HAVAL128
- MHASH_RIPEMD160
- MHASH_GOST
- MHASH_TIGER
- MHASH_CRC32
- MHASH_CRC32B

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4)

Získat název zadaného hashe

```
string mhash_get_hash_name (int hash)
```

mhash_get_hash_name() se používá ke zjištění názvu zadaného hashe.

mhash_get_hash_name() přijímá id hashe jako argument a vrací název tohoto hashe nebo `false`, pokud tento hash neexistuje.

Příklad 1. Ukázka mhash_get_hash_name()

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name ($hash);
?>
```

Výše uvedená ukázka vytiskne:

```
MD5
```

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4)

Získat velikost bloku určeného hashe

```
int mhash_get_block_size (int hash)
```

mhash_get_block_size() se používá ke zjištění velikosti bloku argumentu *hash*.

mhash_get_block_size() přijímá jeden argument, *hash* a vrací velikost v bytech nebo `false`, pokud *hash* neexistuje.

mhash_count (PHP 3>= 3.0.9, PHP 4)

Získat nejvyšší dostupné hash id

```
int mhash_count (void)
```

mhash_count() vrací nejvyšší dostupné hash id. Hashe jsou číslované od 0 po toto hash id.

Příklad 1. Traversing all hashes

```
<?php

$nr = mhash_count();

for ($i = 0; $i <= $nr; $i++) {
    echo sprintf ("The blocksize of %s is %d\n",
        mhash_get_hash_name ($i),
        mhash_get_block_size ($i));
}
?>
```

mhash (PHP 3>= 3.0.9, PHP 4)

Spočítat hash

```
string mhash (int hash, string data, string [ key ])
```

mhash() aplikuje hash funkci určenou argumentem *hash* na *data* a vrací výsledný hash (také nazývaný digest). Pokud je předán *key*, vrací výsledný HMAC. HMAC is keyed hashing for message authentication, or simply a message digest that depends on the specified key. Not all algorithms supported in mhash can be used in HMAC mode. In case of an error returns false.

mhash_keygen_s2k (PHP 4 >= 4.0.4)

Vygenerovat klíč

```
string mhash_keygen_s2k (int hash, string password, string salt, int bytes)
```

mhash_keygen_s2k() generuje klíč, který je *bytes* dlouhý, z předaného hesla. Toto je Salted S2K algoritmus specifikovaný v OpenPGP dokumentu (RFC 2440). Tento algoritmus použije k vytvoření klíče *hash* algoritmus. *salt* musí být pro každý generovaný klíč jiný a dostatečně náhodný, aby vytvořil různé klíče. Salt musí být při kontrole klíčů znám, tudíž je dobrý nápad ho připojit ke klíči. Salt ma pevnou délku 8 bytů a pokud dodáte méně bytů, bude doplněn nulami. Pamatujte, že uživatelsky určená hesla nejsou vhodná k použití jako klíče, protože uživatelé obvykle volí klíče, které mohou napsat na klávesnici. Tato hesla využívají pouze 6 až 7 bytů na znak (nebo méně). Je velmi vhodné na uživateli určené klíče použít nějakou transformaci (jako je tato funkce).

XLIII. Microsoft SQL Server functions

mssql_close (PHP 3, PHP 4)

Close MS SQL Server connection

```
int mssql_close ([int link_identifier])
```

Returns: true on success, false on error.

Mssql_close() closes the link to a MS SQL Server database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Mssql_close() will not close persistent links generated by **mssql_pconnect()**.

See also: **mssql_connect()**, **mssql_pconnect()**.

mssql_connect (PHP 3, PHP 4)

Open MS SQL server connection

```
int mssql_connect ([string servername [, string username [, string password]]])
```

Returns: A positive MS SQL link identifier on success, or false on error.

Mssql_connect() establishes a connection to a MS SQL server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **mssql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mssql_close()**.

See also **mssql_pconnect()**, **mssql_close()**.

mssql_data_seek (PHP 3, PHP 4)

Move internal row pointer

```
int mssql_data_seek (int result_identifier, int row_number)
```

Returns: true on success, false on failure.

Mssql_data_seek() moves the internal row pointer of the MS SQL result associated with the specified result identifier to point to the specified row number. The next call to **mssql_fetch_row()** would return that row.

See also: **mssql_data_seek()**.

mssql_fetch_array (PHP 3, PHP 4)

Fetch row as array

```
int mssql_fetch_array (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

Mssql_fetch_array() is an extended version of **mssql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **mssql_fetch_array()** is NOT significantly slower than using **mssql_fetch_row()**, while it provides a significant added value.

For further details, also see **mssql_fetch_row()**.

mssql_fetch_field (PHP 3, PHP 4)

Get field information

```
object mssql_fetch_field (int result [, int field_offset])
```

Returns an object containing field information.

Mssql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mssql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric

See also **mssql_field_seek()**.

mssql_fetch_object (PHP 3, PHP 4)

Fetch row as object

```
int mssql_fetch_object (int result)
```

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

Mssql_fetch_object() is similar to **mssql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **mssql_fetch_array()**, and almost as quick as **mssql_fetch_row()** (the difference is insignificant).

See also: **mssql_fetch-array()** and **mssql_fetch-row()**.

mssql_fetch_row (PHP 3, PHP 4)

Get row as enumerated array

```
array mssql_fetch_row (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

Mssql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mssql_fetch_rows()** would return the next row in the result set, or false if there are no more rows.

See also: **mssql_fetch_array()**, **mssql_fetch_object()**, **mssql_data_seek()**, **mssql_fetch_lengths()**, and **mssql_result()**.

mssql_field_length (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Get the length of a field

```
int mssql_field_length (int result [, int offset])
```

mssql_field_name (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Get the name of a field

```
int mssql_field_name (int result [, int offset])
```

mssql_field_seek (PHP 3, PHP 4)

Set field offset

```
int mssql_field_seek (int result, int field_offset)
```

Seeks to the specified field offset. If the next call to **mssql_fetch_field()** won't include a field offset, this field would be returned.

See also: **mssql_fetch_field()**.

mssql_field_type (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Get the type of a field

```
string mssql_field_type (int result [, int offset])
```

mssql_free_result (PHP 3, PHP 4)

Free result memory

```
int mssql_free_result (int result)
```

mssql_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **mssql_free_result()** with the result identifier as an argument and the associated result memory will be freed.

mssql_get_last_message (PHP 3, PHP 4)

Returns the last message from server (over min_message_severity?)

```
string mssql_get_last_message  (void )
```

mssql_min_error_severity (PHP 3, PHP 4)

Sets the lower error severity

```
void mssql_min_error_severity  (int severity)
```

mssql_min_message_severity (PHP 3, PHP 4)

Sets the lower message severity

```
void mssql_min_message_severity  (int severity)
```

mssql_num_fields (PHP 3, PHP 4)

Get number of fields in result

```
int mssql_num_fields (int result)
```

Mssql_num_fields() returns the number of fields in a result set.

See also: **mssql_db_query()**, **mssql_query()**, **mssql_fetch_field()**, and **mssql_num_rows()**.

mssql_num_rows (PHP 3, PHP 4)

Get number of rows in result

```
int mssql_num_rows (string result)
```

Mssql_num_rows() returns the number of rows in a result set.

See also: **mssql_db_query()**, **mssql_query()**, and **mssql_fetch_row()**.

mssql_pconnect (PHP 3, PHP 4)

Open persistent MS SQL connection

```
int mssql_pconnect ([string servername [, string username [, string password]]])
```

Returns: A positive MS SQL persistent link identifier on success, or false on error.

Mssql_pconnect() acts very much like **mssql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**mssql_close()** will not close links established by **mssql_pconnect()**).

This type of links is therefore called 'persistent'.

mssql_query (PHP 3, PHP 4)

Send MS SQL query

```
int mssql_query (string query [, int link_identifier])
```

Returns: A positive MS SQL result identifier on success, or false on error.

Mssql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **mssql_connect()** was called, and use it.

See also: **mssql_db_query()**, **mssql_select_db()**, and **mssql_connect()**.

mssql_result (PHP 3, PHP 4)

Get result data

```
int mssql_result (int result, int i, mixed field)
```

Mssql_result() returns the contents of one cell from a MS SQL result set. The field argument can be the field's offset, the field's name or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), it uses the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mssql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **mssql_fetch_row()**, **mssql_fetch_array()**, and **mssql_fetch_object()**.

mssql_select_db (PHP 3, PHP 4)

Select MS SQL database

```
int mssql_select_db (string database_name [, int link_identifier])
```

Returns: true on success, false on error

Mssql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mssql_connect()** was called, and use it.

Every subsequent call to **mssql_query()** will be made on the active database.

See also: **mssql_connect()**, **mssql_pconnect()**, and **mssql_query()**

XLIV. Ming functions for Flash

Introduction

Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms–the only thing that's missing is sound events.

Ming is not an acronym.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net (mailto:ming@opaque.net).

Ming was added in PHP 4.0.5.

Installation

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page : http://www.opaque.net/ming/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the `PREFIX=` line in the `Makefile` to change the installation directory.

built into php (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure –with-ming <other config options>
```

Build and install php as usual, Restart web server if necessary

built into php (unix)

download `php_ming.so.gz`. uncompress it and copy it to your php modules directory. (you can find your php module directory by running **php-config –extension-dir**). Now either just add `extension=php_ming.so` to your `php.ini` file, or put `dl('php_ming.so');` at the head of all of your Ming scripts.

How to use Ming

Ming introduces 13 new object in PHP, all with matching methods and attributes. To use them, you need to know about

objects.

- **swfmovie()**.
- **swfshape()**.
- **swfdisplayitem()**.
- **swfgradient()**.
- **swfbitmap()**.
- **swffill()**.
- **swfmorph()**.
- **swftext()**.
- **swffont()**.
- **swftextfield()**.
- **swfsprite()**.
- **swfbutton()**.
- **swfaction()**.

SWFMovie (PHP 4 CVS only)

Creates a new movie object, representing an SWF version 4 movie.

```
new swfmovie (void)
```

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** et **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFMovie->output (unknown)

Dumps your lovingly prepared movie out.

```
void swfmovie->output (void)
```

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, **swfaction()**... Any example will use this method.

SWFMovie->save (unknown)

Saves your movie in a file.

```
void swfmovie->save (string filename)
```

swfmovie->save() saves your movie to the file named *filename*.

See also **output()**

SWFMovie->add (unknown)

Adds any type of data to a movie.

```
void swfmovie->add (ressource instance)
```

swfmovie->add() adds *instance* to the current movie. *instance* is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->remove (unknown)

Removes the object instance from the display list.

```
void swfmovie->remove (ressource instance)
```

swfmovie->remove() removes the object instance *instance* from the display list.

See also **swfmovie->add()**

SWFMovie->setbackground (unknown)

Sets the background color.

```
void swfmovie->setbackground (int red, int green, int blue)
```

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setrate (unknown)

Sets the animation's frame rate.

```
void swfmovie->setrate (int rate)
```

nom_de_la_fonction() sets the frame rate to *rate*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->setdimension (unknown)

Sets the movie's width and height.

```
void swfmovie->setdimension (int width, int height)
```

swfmovie->setdimension() sets the movie's width to *width* and height to *height*.

SWFMovie->setframes (unknown)

Sets the total number of frames in the animation.

```
void swfmovie->setframes (string numberofframes)
```

swfmovie->setframes() sets the total number of frames in the animation to *numberofframes*.

SWFMovie->nextframe (unknown)

Moves to the next frame of the animation.

```
void swfmovie->nextframe (void)
```

swfmovie->setframes() moves to the next frame of the animation.

SWFMovie->streammp3 (unknown)

Streams a MP3 file.

```
void swfmovie->streammp3 (string mp3FileName)
```

swfmovie->streammp3() streams the mp3 file *mp3FileName*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Příklad 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
// use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem (unknown)

Creates a new displayitem object.

```
new swfdisplayitem (void)
```

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** et **swfdisplayitem->multcolor()**.

SWFDisplayItem->moveTo (unknown)

Moves object in global coordinates.

```
void swfdisplayitem->moveto (int x, int y)
```

swfdisplayitem->moveto() moves the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->move (unknown)

Moves object in relative coordinates.

```
void swfdisplayitem->move (int dx, int dy)
```

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->scaleTo (unknown)

Scales the object in global coordinates.

```
void swfdisplayitem->scaleto (int x, int y)
```

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->scale (unknown)

Scales the object in relative coordinates.

```
void swfdisplayitem->scale (int dx, int dy)
```

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->rotateTo (unknown)

Rotates the object in global coordinates.

```
void swfdisplayitem->rotateto (double degrees)
```

swfdisplayitem->rotateto() set the current object rotation to *`degrees`* degrees in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Příklad 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal php var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
    // return the changed object.  kinda weird..

    return $i;
 }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
```

```
  }

  // see?  it sure paid off in legibility:

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->Rotate (unknown)

Rotates in relative coordinates.

```
void swfdisplayitem->rotate (double ddegrees)
```

swfdisplayitem->rotate() rotates the current object by *ddegrees* degrees from its current rotation.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->skewXTo (unknown)

Sets the X-skew.

```
void swfdisplayitem->skewxto (double degrees)
```

swfdisplayitem->skewxto() sets the x-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewX (unknown)

Sets the X-skew.

```
void swfdisplayitem->skewx (double ddegrees)
```

swfdisplayitem->skewx() adds *ddegrees* to current x-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewYTo (unknown)

Sets the Y-skew.

```
void swfdisplayitem->skewyto (double degrees)
```

swfdisplayitem->skewyto() sets the y-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewY (unknown)

Sets the Y-skew.

```
void swfdisplayitem->skewy (double ddegrees)
```

swfdisplayitem->skewy() adds *ddegrees* to current y-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->setDepth (unknown)

Sets z-order

```
void swfdisplayitem->setdepth (double depth)
```

swfdisplayitem->rotate() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->remove (unknown)

Removes the object from the movie

```
void swfdisplayitem->remove (void)
```

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Sets the object's name

```
void swfdisplayitem->setname (string name)
```

swfdisplayitem->setname() sets the object's name to *name*, for targetting with action script. Only useful on sprites.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Sets the object's ratio.

```
void swfdisplayitem->setratio (double ratio)
```

swfdisplayitem->setratio() sets the object's ratio to *ratio*. Obviously only useful for morphs.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Příklad 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
```

```
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor (unknown)

Adds the given color to this item's color transform.

```
void swfdisplayitem->addcolor ([integer red [, integer green [, integer blue [, integer a]]]])
```

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->multColor (unknown)

Multiplies the item's color transform.

```
void swfdisplayitem->multcolor ([integer red [, integer green [, integer blue [, integer a]]]])
```

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Příklad 1. swfdisplayitem->multcolor() example

```
<?php
```

```
  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape (PHP 4 CVS only)

Creates a new shape object.

```
new swfshape (void)
```

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfile()**, **swfshape->setrightfile()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** et **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Příklad 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLine (unknown)

Sets the shape's line style.

```
void swfshape->setline (int width [, integer red [, integer green [, integer blue [,
integer a]]]])
```

swfshape->setline() sets the shape's line style. *width* is the line's width. If *width* is 0, the line's style is removed (then, all other arguments are ignored). If *width* > 0, then line's color is set to *red*, *green*, *blue*. Last parameter *a* is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Příklad 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
```

```
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->addFill (unknown)

Adds a solid fill to the shape.

```
void swfshape->addfill (integer red, integer green, integer blue [, integer a])
```

```
void swfshape->addfill (SWFbitmap bitmap [, integer flags])
```

```
void swfshape->addfill (SWFGradient gradient [, integer flags])
```

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an **swfbitmap()** object. The *flags* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *gradient* argument is an **swfgradient()** object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an **swffill()** object for use with the **swfshape->setfill()** functions described below.

See also **swfshape->setfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Příklad 1. swfshape->addfill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // use your own bitmap
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);
```

```
  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLeftFill (unknown)

Sets left rasterizing color.

```
void swfshape->setleftfill (swfgradient fill)
```

```
void swfshape->setleftfill (integer red, integer green, integer blue [, integer a])
```

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setRightFill (unknown)

Sets right rasterizing color.

```
void swfshape->setrightfill (swfgradient fill)
```

```
void swfshape->setrightfill (integer red, integer green, integer blue [, integer a])
```

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape->movePenTo (unknown)

Moves the shape's pen.

```
void swfshape->movepento (integer x, integer y)
```

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** et **swfshape->drawline()**.

SWFShape->movePen (unknown)

Moves the shape's pen (relative).

```
void swfshape->movepen (integer dx, integer dy)
```

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** et **swfshape->drawline()**.

SWFShape->drawLineTo (unknown)

Draws a line.

```
void swfshape->drawlineto (integer x, integer y)
```

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** et **swfshape->drawline()**.

SWFShape->drawLine (unknown)

Draws a line (relative).

```
void swfshape->drawline (integer dx, integer dy)
```

swfshape->setrightfill() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** et **swfshape->drawlineto()**.

SWFShape->drawCurveTo (unknown)

Draws a curve.

```
void swfshape->drawcurveto (integer controlx, integer controly, integer anchorx,
integer anchory)
```

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*anchorx*,*anchory*) using (*controlx*,*controly*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** et **swfshape->movepen()**.

SWFShape->drawCurve (unknown)

Draws a curve (relative).

```
void swfshape->drawcurve (integer controldx, integer controldy, integer anchordx,
integer anchordy)
```

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*anchorx*,*anchory*) using relative control point (*controlx*,*controly*). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** et **swfshape->movepen()**.

SWFGradient (PHP 4 CVS only)

Creates a gradient object

```
new swfgradient (void)
```

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Příklad 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
```

```
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry (unknown)

Adds an entry to the gradient list.

```
void swfgradient->addentry (double ratio, integer red, integer green, integer blue [,
integer a])
```

swfgradient->addentry() adds an entry to the gradient list. *ratio* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

SWFBitmap (PHP 4 CVS only)

Loads Bitmap object

```
new swfbitmap (string filename [, integer alphafilename])
```

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named *filename*. *alphafilename* indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Poznámka:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can’t import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don’t want a dependency on the png library in ming- autoconf should solve this, but that’s not set up yet.

Příklad 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Příklad 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
```

```
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFBitmap->getWidth (unknown)

Returns the bitmap's width.

```
int swfbitmap->getwidth (void)
```

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap->getHeight (unknown)

Returns the bitmap's height.

```
int swfbitmap->getheight (void)
```

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFFill (PHP 4 CVS only)

Loads SWFFill object

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

SWFFill->moveTo (unknown)

Moves fill origin

```
void swffill->moveto (integer x, integer y)
```

swffill->moveto() moves fill's origin to (*x*,*y*) in global coordinates.

SWFFill->scaleTo (unknown)

Sets fill's scale

```
void swffill->scaleto (integer x, integer y)
```

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->rotateTo (unknown)

Sets fill's rotation

```
void swffill->rotateto (double degrees)
```

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->skewXTo (unknown)

Sets fill x-skew

```
void swffill->skewxto (double x)
```

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo (unknown)

Sets fill y-skew

```
void swffill->skewyto (double y)
```

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFMorph (PHP 4 CVS only)

Creates a new SWFMorph object.

```
new swfmorph (void)
```

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Příklad 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Gets a handle to the starting shape

```
mixed swfmorph->getshape1 (void)
```

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an **swfshape()** object.

SWFMorph->getshape2 (unknown)

Gets a handle to the ending shape

```
mixed swfmorph->getshape2 (void)
```

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an **swfshape()** object.

SWFMorph (PHP 4 CVS only)

Creates a new SWFText object.

```
new swftext (void)
```

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Příklad 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->setFont (unknown)

Sets the current font

```
void swftext->setfont (string font)
```

swftext->setfont() sets the current font to *font*.

SWFText->setHeight (unknown)

Sets the current font height

```
void swftext->setheight (integer height)
```

swftext->setheight() sets the current font height to *height*. Default is 240.

SWFText->setSpacing (unknown)

Sets the current font spacing

```
void swftext->setspacing (double spacing)
```

swftext->setspacing() sets the current font spacing to *spacingspacing*. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText->setColor (unknown)

Sets the current font color

```
void swftext->setcolor (integer red, integer green, integer blue [, integer a])
```

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->moveTo (unknown)

Moves the pen

```
void swftext->moveto (integer x, integer y)
```

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->addString (unknown)

Draws a string

```
void swftext->addstring (string string)
```

swftext->addstring() draws the string *string* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth (unknown)

Computes string's width

```
void swftext->addstring (string string)
```

swftext->addstring() returns the rendered width of the *string* string at the text object's current font, scale, and spacing settings.

SWFFont (PHP 4 CVS only)

Loads a font definition

```
new swffont (string filename)
```

If *filename* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

getwidth (PHP 4 CVS only)

Returns the string's width

```
int swffont->getwidth (string string)
```

swffont->getwidth() returns the string *string*'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFTextField (PHP 4 CVS only)

Creates a text field object

```
new swftextfield ([int flags])
```

swftextfield() creates a new text field object. Text Fields are less flexible than **swftext()** objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_WORDWRAP allows text to wrap
- SWFTEXTFIELD_NOSELECT makes the field non-selectable

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** et **swftextfield->addstring()**.

SWFTextField->setFont (unknown)

Sets the text field font

```
void swftextfield->setfont (string font)
```

swftextfield->setfont() sets the text field font to the [browser-defined?] *font* font.

SWFTextField->setbounds (unknown)

Sets the text field width and height

```
void swftextfield->setbounds (int width, int height)
```

swftextfield->setbounds() sets the text field width to *width* and height to *height*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->align (unknown)

Sets the text field alignment

```
void swftextfield->align (int alignement)
```

swftextfield->align() sets the text field alignment to *alignement*. Valid values for *alignement* are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setHeight (unknown)

Sets the font height of this text field font.

```
void swftextfield->setheight (int height)
```

swftextfield->setheight() sets the font height of this text field font to the given height *height*. Default is 240.

SWFTextField->setLeftMargin (unknown)

Sets the left margin width of the text field.

```
void swftextfield->setleftmargin (int width)
```

swftextfield->setleftmargin() sets the left margin width of the text field to *width*. Default is 0.

SWFTextField->setrightMargin (unknown)

Sets the right margin width of the text field.

```
void swftextfield->setrightmargin (int width)
```

swftextfield->setrightmargin() sets the right margin width of the text field to *width*. Default is 0.

SWFTextField->setMargins (unknown)

Sets the margins width of the text field.

```
void swftextfield->setmargins (int left, int right)
```

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setindentation (unknown)

Sets the indentation of the first line.

```
void swftextfield->setindentation (int width)
```

swftextfield->setindentation() sets the indentation of the first line in the text field, to *width*.

SWFTextField->setLineSpacing (unknown)

Sets the line spacing of the text field.

```
void swftextfield->setlinespacing (int height)
```

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *height*. Default is 40.

SWFTextField->setcolor (unknown)

Sets the color of the text field.

```
void swftextfield->setcolor (integer red, integer green, integer blue [, integer a])
```

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setname (unknown)

Sets the variable name

```
void swftextfield->setname (string name)
```

swftextfield->setname() sets the variable name of this text field to *`name`*, for form posting and action scripting purposes.

SWFTextField->addstring (unknown)

Concatenates the given string to the text field

```
void swftextfield->addstring (string string)
```

swftextfield->setname() concatenates the string *`string`* to the text field.

SWFSprite (PHP 4 CVS only)

Creates a movie clip (a sprite)

```
new swfsprite (void)
```

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** et **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Příklad 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
```

```
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add (unknown)

Adds an object to a sprite

```
void swfsprite->add (ressource object)
```

swfsprite->add() adds a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object.

For displayable types (**swfshape()**, **swfbutton()**, **swftext()**, **swfaction()** or **swfsprite()**), this returns a handle to the object in a display list.

SWFSprite->remove (unknown)

Removes an object to a sprite

```
void swfsprite->remove (ressource object)
```

swfsprite->remove() remove a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object from the sprite.

SWFSprite->setframes (unknown)

Sets the total number of frames in the animation.

```
void swfsprite->setframes (integer numberofframes)
```

swfsprite->setframes() sets the total number of frames in the animation to *numberofframes*.

SWFSprite->nextframe (unknown)

Moves to the next frame of the animation.

```
void swfsprite->nextframe (void)
```

swfsprite->setframes() moves to the next frame of the animation.

SWFbutton (PHP 4 CVS only)

Creates a new Button.

```
new swfbutton (void)
```

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover()** **swfbutton->setdown()**, **swfbutton->sethit()** **swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Příklad 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
  function addLabel($string)
  {
    global $p;

    $i = $p->add(label($string));
    $p->nextFrame();
    $p->remove($i);
  }

  $p->add(new SWFAction("stop();"));
  addLabel("NO ACTION");
  addLabel("SWFBUTTON_MOUSEUP");
  addLabel("SWFBUTTON_MOUSEDOWN");
  addLabel("SWFBUTTON_MOUSEOVER");
  addLabel("SWFBUTTON_MOUSEOUT");
  addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
  addLabel("SWFBUTTON_DRAGOVER");
  addLabel("SWFBUTTON_DRAGOUT");

  function rect($r, $g, $b)
  {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(600,0);
    $s->drawLine(0,600);
    $s->drawLine(-600,0);
    $s->drawLine(0,-600);

    return $s;
  }

  $b = new SWFButton();
```

```
  $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
  $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
  $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
        SWFBUTTON_MOUSEUP);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
SWFBUTTON_DRAGOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
SWFBUTTON_DRAGOUT);

  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Příklad 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b-
>addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
```

```
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFbutton->addShape (unknown)

Adds a shape to a button

```
void swfbutton->addshape (ressource shape, integer flags)
```

swfbutton->addshape() adds the shape *shape* to this button. The following *flags*' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setUp (unknown)

Alias for addShape(shape, SWFBUTTON_UP)

```
void swfbutton->setup (ressource shape)
```

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

SWFbutton->setOver (unknown)

Alias for addShape(shape, SWFBUTTON_OVER)

```
void swfbutton->setover (ressource shape)
```

swfbutton->setup() alias for addShape(shape, SWFBUTTON_OVER).

SWFbutton->setHit (unknown)

Alias for addShape(shape, SWFBUTTON_HIT)

```
void swfbutton->sethit (ressource shape)
```

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

SWFbutton->setAction (unknown)

Sets the action

```
void swfbutton->setaction (ressource action)
```

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). *`action`* is a **swfaction()**.

SWFAction (PHP 4 CVS only)

Creates a new Action.

```
new swfaction (string script)
```

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (false) or 1 (true). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given url into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie). The optional method argument can be post or get if you want to submit variables back to the server.

loadMovie(url, target)

Same as above, more or less. Come to think of it, I don't quite know what the difference is.

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Příklad 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
```

```
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Příklad 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));


  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();
```

```
$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);


/* sprite container for morphing box -
   this is just a timeline w/ the box morphing */

$box = new SWFSprite();
$box->add(new SWFAction("
  stop();
"));
$i = $box->add($r);

for($n=0; $n<=20; ++$n)
{
  $i->setRatio($n/20);
  $box->nextFrame();
}


/* this container sprite allows us to use the same action code many times */

$cell = new SWFSprite();
$i = $cell->add($box);
$i->setName('box');

$cell->add(new SWFAction("

  setTarget('box');

  /* ...x means the x coordinate of the parent, i.e. (..).x */
  dx = (/mouse.x + random(6)-3 - ...x)/5;
  dy = (/mouse.y + random(6)-3 - ...y)/5;
  gotoFrame(int(dx*dx + dy*dy));

"));

$cell->nextFrame();
$cell->add(new SWFAction("

  gotoFrame(0);
  play();

"));

$cell->nextFrame();


/* finally, add a bunch of the cells to the movie */

for($x=0; $x<12; ++$x)
{
  for($y=0; $y<8; ++$y)
  {
    $i = $m->add($cell);
    $i->moveTo(100*$x+50, 100*$y+50);
  }
}

$m->nextFrame();
```

```
  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Příklad 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
```

```
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));
```

```
  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Příklad 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
   "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }


  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);


  /* button checks for pressed key, sends sprite to the right frame */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
```

```
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

XLV. Smíšené funkce

Tyto funkce byly umístěny zde, protože nespadají do žádné jiné kategorie.

connection_aborted (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Vrací true, pokud se klient odpojil

```
int connection_aborted (void )
```

Vrátí true, pokud se klient odpojil. Úplné vysvětlení viz popis v Obsluha spojení v kapitole Vlastnosti.

connection_status (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Vrací bitové pole stavu spojení

```
int connection_status (void )
```

Vrací bitové pole stavu spojení. Úplné vysvětlení viz popis v Obsluha spojení v kapitole Vlastnosti.

connection_timeout (PHP 3>= 3.0.7, 4.0b4 - 4.0.4 only)

Return true if script timed out

```
int connection_timeout (void )
```

Returns true if script timed out. Úplné vysvětlení viz popis v Obsluha spojení v kapitole Vlastnosti.

define (PHP 3, PHP 4)

Definuje pojmenovanou konstantu.

```
int define (string name, mixed value [, int case_insensitive])
```

Definuje pojmenovanou konstantu, která je podobná proměnné s výjimkou toho, že:

- Konstanty nemají znak dolaru ('$') před jménem;
- Konstanty jsou dostupné odkudkoliv bez ohledu na pravidla rozsahu platnosti proměnných;
- Konstanty se nedají předefinovávat a rušit; a
- Konstanty se mohou nabývat pouze skalárních hodnot.

Název konstanty je dán argumentem *name*; hodnota je dána argumentem *value*.

Dále je dostupný volitelní třetí argument *case_insensitive*. Pokud má hodnotu *1*, konstanta bude definována case-insensitive. Výchozí chování je case-sensitive; tj. CONSTANT and Constant reprezentují různé hodnoty.

Příklad 1. Definování konstant

```
<?php
define ("CONSTANT", "Hello world.");
echo CONSTANT; // výstup je "Hello world."
?>
```

Define() vrací TRUE při úspěchu a FALSE pokud dojde k chybě.

Viz také **defined()** a sekci Konstanty.

defined (PHP 3, PHP 4)

Zkontroluje, jestli existuje daná pojmenovaná konstanta

```
int defined (string name)
```

Vrací true, pokud byla definována pojmenovaná konstanta daná argumentem *name*, jinak false.

Příklad 1. Kontrola konstant

```
<?php
if (defined("CONSTANT")){ // název konstanty by měl být v uvozovkách
    echo CONSTANT; //
    }
?>
```

Viz také **define()** a sekci Konstanty.

die (unknown)

Vytiskne vzkaz a ukončí současný skript

```
void die (string message)
```

Tento jazykový konstrukt vytiskne vzkaz a ukončí parsování skriptu. Nemá návratovou hodnotu.

Příklad 1. die example

```
<?php
$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
    or die("nepodařilo se otevřít soubor ($filename)");
?>
```

Viz také **exit()**.

eval (unknown)

Vyhodnotí řetězec jako PHP kód

```
mixed eval (string code_str)
```

eval() vyhodnotí řetězec předaný v *code_str* jako PHP kód. Kromě jiného se dá využít na ukládání kódu v textovém sloupci databáze pro pozdější vykonání.

Při používání **eval()** byste měli mít na paměti několik faktorů. Pamatujte si, že předávaný řetězec musí být platný PHP kód, včetně věcí jako ukončování výrazů středníkem, aby parser nezemřel na řádku po **eval()**, a řádné escapování v *code_str*.

Také pamatujte, ze hodnoty přiřazené proměnným v **eval()** těmto proměnným zůstanou i v hlavním skriptu.

Výraz `return` okamžitě ukončí vyhodnocování předaného řetězce. V PHP 4 můžete použít `return` k vrácení hodnoty, která se stane výsledkem **eval()** funkce, zatímco v PHP 3 byl **eval()** typu `void` nic nevracel.

Příklad 1. Eval() příklad - jednoduché spojení textů

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval ("\$str = \"$str\";");
echo $str;
?>
```

Výše uvedený příklad ukáže:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

exit (unknown)

Ukončí současný skript

```
void exit(void);
```

Tento jazykový konstrkut ukončí parsování skriptu. Nevrací žádnou hodnotu.

Viz také **die()**.

get_browser (PHP 3, PHP 4)

Určuje schopnosti uživatelova browseru

```
object get_browser ([string user_agent])
```

get_browser() se pokusí určit schopnosti uživatelova browseru. Toho je dosaženo vyhledáním informací o browseru v souboru `browscap.ini`. Standardne se použije $HTTP_USER_AGENT; nicméně, můžete to změnit (tj. vyhledat informace o jiném browseru) předáním volitelného argumentu *`user_agent`*.

Informace se vracejí jako objekt, který obsahuje různé datové elementy, které reprezentují například hlavní a vedlejší číslo verze a ID řetězec; true/false hodnoty vlastností jako podpora rámců, JavaScript a cookies, atd.

Jakkoli `browscap.ini` obsahuje informace o mnoha browserech, aktuálnost databáze závisí na uživatelských updatech. Formát souboru je poměrně snadno pochopitelný.

Následující příklad ukazuje, jak by se daly vypsat všechny informace získané o uživatelově browseru.

Příklad 1. Get_browser() příklad

```
<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
    $str .= "<b>$key:</b> $value<br>\n";
    }
    return $str;
}
```

```
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>
```

Výstup z výše uvedeného skriptu by vypadal asi takto:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br>
<b>parent:</b> Netscape 4.0<br>
<b>platform:</b> Unknown<br>
<b>majorver:</b> 4<br>
<b>minorver:</b> 5<br>
<b>browser:</b> Netscape<br>
<b>version:</b> 4<br>
<b>frames:</b> 1<br>
<b>tables:</b> 1<br>
<b>cookies:</b> 1<br>
<b>backgroundsounds:</b> <br>
<b>vbscript:</b> <br>
<b>javascript:</b> 1<br>
<b>javaapplets:</b> 1<br>
<b>activexcontrols:</b> <br>
<b>beta:</b> <br>
<b>crawler:</b> <br>
<b>authenticodeupdate:</b> <br>
<b>msn:</b> <br>
```

Aby to fungovalo, browscap direktiva ve vašem konfiguračním souboru musí ukazovat na platné umístění `browscap.ini` souboru.

Pro další informace (včetně lokací na kterých můžete získat `browscap.ini` soubor) viz PHP FAQ na http://www.php.net/FAQ.php.

highlight_file (PHP 4)

Zvýrazní syntaxi souboru

```
boolean highlight_file (string filename)
```

Funkce **highlight_file()** vytiskne barevně zvýrazněnou syntaxi kódu obsaženého ve *`filename`* s použitím barev definovaných ve zvýrazňovači syntaxe zabudovaném v PHP. Vrací true při úspěchu, jinak false (PHP 4).

Příklad 1. Tvorba URL zvyrazňující syntaxi

K vytvoření URL, která zvýrazní syntaxi jakéhokoliv skriptu, který jí předáte využijeme "ForceType" direktivu Apache k vytvoření hezkého vzorce URL, and pomocí funkce **highlight_file()** vypíšeme hezky vypadající výpis kódu.

Do svého httpd.conf přidejte následující:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

A potom vytvořte soubor pojmenovaný "source", a umístěte ho do svého web root adresáře.

```
<HTML>
<HEAD>
```

```
<TITLE>Zobrazení zdroje</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>CHYBA: Je potřeba název skriptu!</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Zdroj souboru: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>CHYBA: Povoleny jsou pouze soubory s příponou .php nebo .inc!</H1>";
    }
    }
    echo "<HR>Zpracováno: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Potom můžete použít URL jako je ta níže k zobrazení obarvené verze skriptu umístěné v "/path/to/script.php" na vašem webu.

```
http://your.server.com/source/path/to/script.php
```

Viz také **highlight_string()**, **show_source()**.

highlight_string (PHP 4)

Zvýraznění syntaxe řetězce

```
void highlight_string (string str)
```

Funkce **highlight_string()** vytiskne barevně zvýrazněnou verzi *str* s použitím barev definovaných ve zvýrazňovači syntaxe zabudovaném v PHP. Vrací true při úspěchu, jinak false (PHP 4).

Viz také **highlight_file()**, **show_source()**.

ignore_user_abort (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Nastavuje, jestli má ukončení spojení klientem přerušit vykonávání skriptu

```
int ignore_user_abort ([int setting])
```

Tato funkce nastavuje, jestli má odpojení klienta způsobit ukončení skriptu. Vrací předchozí nastavení, a při zavolání bez argumentu současné nastavení nemění, pouze ho vrací. Úplné vysvětlení viz popis v sekci Obsluha spojení v kapitole Vlastnosti

iptcparse (PHP 3>= 3.0.6, PHP 4)

Parsuje binární IPTC http://www.iptc.org/ blok do jednotlivých tagů.

```
array iptcparse (string iptcblock)
```

Tato funkce parsuje binární IPTC blok do jeho jednotlivých tagů. Vrací pole, které používá tagmarker jako index, a jeho hodnotu jako hodnotu. Vrací false při chybě, nebo pokud nenajde žádná IPTC data. Příklad viz **GetImageSize()**.

leak (PHP 3, PHP 4)

Nenávratně alokuje pamět

```
void leak (int bytes)
```

Leak() nenávratně alokuje specifikované množství paměti.

Tato funkce je užitečná při ladění správce paměti, který automaticky čistí "vyteklou" (leaked) pamět při dokončení každého požadavku.

pack (PHP 3, PHP 4)

Sbalí data do binárního řetězce.

```
string pack (string format [, mixed args ...])
```

Sbalí předané argumenty do binárního řetězce podle argumentu *format*. Vrací binární řetězec obsahující předaná data.

Nápad na tuto funkci byl převzat z Perlu, a všechny formátovací kódy fungují stejně jako tam, nicméně, některé formátovací kódy chybí, jako například Perlovský formátovací kód "u". Formátovací řetězec sestává z formátovacích kódu následovaných volitelným opakovacím argumentem. Opakovací argument může být buď' celočíselná hodnota, nebo * pro opakování do konce vstupních dat. U a, A, h, H počet opakování určuje, kolik znaků se vezme z jednoho datového argumentu, u @ je to absolutní pozice, kde se mají umístit další data, u všeho ostatního počet opakování určuje, kolik datových argumentů se spotřebuje a sbalí do výsledného binárního řetězce. V současnosti jsou implementovány

- a řetězec doplněný NUL hodnotami
- A řetězec doplněný SPACE hodnotami
- h Hex řetězec, spodní slabika první
- H Hex řetězec, horní slabika první
- c signed char
- C unsigned char
- s signed short (vždy 16 bitů, machine byte order)
- S unsigned short (vždy 16 bitů, machine byte order)
- n unsigned short (vždy 16 bitů, big endian byte order)
- v unsigned short (vždy 16 bitů, little endian byte order)
- i signed integer (velikost a pořadí bytů závislá na systému)
- I unsigned integer (velikost a pořadí bytů závislá na systému)
- l signed long (vždy 32 bitů, machine byte order)
- L unsigned long (vždy 32 bitů, machine byte order)
- N unsigned long (vždy 32 bitů, big endian byte order)
- V unsigned long (vždy 32 bitů, little endian byte order)

- f float (velikost a reprezentace závislá na systému)
- d double (velikost a reprezentace závislá na systému)
- x NUL byte
- X Back up one byte
- @ NUL-fill to absolute position

Příklad 1. Pack() formátovací řetězec

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

Výsledný binární řetězec bude 6 bytů dlouhý, a bude obsahovat bytovou sekvenci 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Všimněte si, že rozdíl mezi hodnotami se znaménkem a bez znaménka ovlivňuje pouze funkci **unpack()**, zatímco funkce **pack()** dává stejný výsledek pro formátovací kódy se znaménkem i bez znaménka.

Dále si všimněte, že PHP interně ukládá celočíselné hodnoty jako hodnoty se znaménkem o velikosti závislé na systému. Pokud zadáte hodnotu bez znaménka, která bude příliš velká, než aby se dala takto uložit, převede se na double, což často vytváří nežádoucí výsledky.

show_source (PHP 4)

Zvýrazní syntaxi souboru

```
boolean show_source (string filename)
```

Funkce **show_source()** vytiskne barevně zvýrazněnou syntaxi kódu obsaženého ve *filename* s použitím barev definovaných ve zvýrazňovači syntaxe zabudovaném v PHP. Vrací true při úspěchu, jinak false (PHP 4).

> **Poznámka:** Tato funkce je alias funkce **highlight_file()**

Viz také **highlight_string()**, **highlight_file()**.

sleep (PHP 3, PHP 4)

Odloží provedení

```
void sleep (int seconds)
```

Funkce sleep odkládá provedení programu o počet sekund předaný v argumentu *seconds*.

Viz také **usleep()**.

uniqid (PHP 3, PHP 4)

Generatuje unikátní id

```
int uniqid (string prefix [, boolean lcg])
```

Uniqid() vrací unikátní identifikátor založený na současném čase v mikrosekundách, opatřený prefixem. Prefix může být užitečný například pokud generujete identifikátory na několika serverech současně, které by mohly vygenerovat identifikátor ve stejnou mikrosekundu. *`Prefix`* může být až 114 znaků dlouhý.

Pokud je volitelný argument *`lcg`* true, **uniqid()** přidá dodatečnou "kombinovanou LCG" entropii na konec své návratové hodnoty, což by mělo učinit výsledky ještě unikátnějšími.

Pokud je *`prefix`* prázdný, vrácený řetěec bude 13 znaků dlouhý. Pokud je *`lcg`* true, bude dlouhý 23 znaků.

> **Poznámka:** *`lcg`* argument je přístupný pouze v PHP 4 and PHP 3.0.13 a vyšších.

Pokud potřebujete unikátní identifikátor nebo symbol, a zamýšlíte předat tento symbol uživateli po síti (např. session cookies), doporučujeme použít něco jako

```
$token = md5 (uniqid ("")); // no random portion
$better_token = md5 (uniqid (rand())); // better, difficult to guess
```

Toto vytvoří 32znakový identifikátor (128bitové hexa číslo), které se extrémně težko předpovídá.

unpack (PHP 3, PHP 4)

Rozbalí data z binárního řetězce

```
array unpack (string format, string data)
```

Unpack() rozbalí data z binárního řetězce do pole podle *`format`*. Vrací pole obsahující rozbalené prvky binárního řetězce.

Unpack() funguje trochu jinak než v Perlu, jelikož rozbalená data jsou uložena v asociativním poli. Toho dosáhnete tak, že vyjmenujete různé formátovací kódy a oddělíte je lomítkem /.

Příklad 1. Unpack() format string

```
$array = unpack ("c2chars/nint", $binarydata);
```

Výsledné pole obsahuej položky "chars1", "chars2" and "int".

Vysvětlení formátovacích kódů viz také: **pack()**

Všimněte si, že PHP interně ukládá celočíselné hodnoty se znaménkem. Pokud rozbalíte velkou celočíselnou hodnotu bez znaménka a ta má stejnou velikost jako hodnoty interně ukládané PHP, výsledkem bude negativní číslo, i když bylo zadáno rozbalování bez znaménka.

usleep (PHP 3, PHP 4)

Odloží provedení v mikrosekundách

```
void usleep (int micro_seconds)
```

Funkce **usleep()** odloží provedení skriptu o daný počet *`micro_seconds`*.

Viz také **sleep()**.

> **Poznámka:** Tato funkce nefunguje na Windows systémech.

XLVI. mnoGoSearch Functions

These functions allow you to access mnoGoSearch (former UdmSearch) free search engine. In order to have these functions available, you must compile php with mnogosearch support by using the `-with-mnogosearch` option. If you use this option without specifying the path to mnogosearch, php will look for mnogosearch under /usr/local/mnogosearch path by default. If you installed mnogosearch at other path you should specify it: `-with-mnogosearch=DIR`.

mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has number of unique features, which makes it appropriate for a wide range of application from search within your site to a specialized search system such as cooking recipes or newspaper search, ftp archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data collected by indexer is used later in a search process. The search is performed via Web interface. C CGI, PHP and Perl search front ends are included.

> **Poznámka:** php contains built-in mysql access library, which can be used to access mysql. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic mysql libraries. Thus, if you use mnoGoSearch with mysql, during php configuration you have to indicate directory of mysql installation, that was used during mnoGoSearch configuration, i.e. for example: `-with-mnogosearch -with-mysql=/usr`

You need at least 3.1.10 version of mnoGoSearch installed to use these functions.

More information about mnoGoSearch can be found at http://www.mnogosearch.ru/.

udm_add_search_limit (PHP 4 CVS only)

Add various search limits

```
int udm_add_search_limit (int agent, int var, string val)
```

udm_add_search_limit() returns TRUE on success, FALSE on error. Adds search restrictions.

agent - a link to Agent, received after call to **udm_alloc_agent()**.

var - defines parameter, indicating limit.

val - defines value of the current parameter.

Possible *var* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain.__/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.
- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.
- UDM_LIMIT_DATE - defines limitation by date document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_alloc_agent (PHP 4 CVS only)

Allocate mnoGoSearch session

```
int udm_alloc_agent (string dbaddr [, string dbmode])
```

udm_alloc_agent() returns mnogosearch agent identifier on success, FALSE on error. This function creates a session with database parameters.

dbaddr - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

dbmode - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requres less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

Poznámka: *dbaddr* and *dbmode* must match those used during indexing.

Poznámka: In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by **udm_find()**.

udm_api_version (PHP 4 CVS only)

Get mnoGoSearch API version.

```
int udm_api_version ()
```

udm_api_version() returns mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return `30110`.

This function allows user to identify which API functions are available, e.g. **udm_get_doc_count()** function is only available in mnoGoSearch 3.1.11 or later.

Example:

```
       if (Udm_Api_Version() >= 30111) {
   print  "Total number of urls in database: ".Udm_Get_Doc_Count($udm)."<br>\n";
}
```

udm_clear_search_limits (PHP 4 CVS only)

Clear all mnoGoSearch search restrictions

```
int udm_clear_search_limits (int agent)
```

udm_clear_search_limits() resets defined search limitations and returns `TRUE`.

udm_errno (PHP 4 CVS only)

Get mnoGoSearch error number

```
int udm_errno (int agent)
```

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving numeric agent error code.

udm_error (PHP 4 CVS only)

Get mnoGoSearch error message

```
string udm_error (int agent)
```

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving agent error message.

udm_find (PHP 4 CVS only)

Perform search

```
int udm_find (int agent, string query)
```

udm_find() returns result link identifier on success, FALSE on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent (PHP 4 CVS only)

Free mnoGoSearch session

```
int udm_free_agent (int agent)
```

udm_free_agent() returns TRUE on success, FALSE on error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Freeing up memory allocated for agent session.

udm_free_ispell_data (PHP 4 CVS only)

Free memory allocated for ispell data

```
int udm_free_ispell_data (int agent)
```

udm_free_ispell_data() always returns TRUE.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

> **Poznámka:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res (PHP 4 CVS only)

Free mnoGoSearch result

```
int udm_free_res (int res)
```

udm_free_res() returns TRUE on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

Freeing up memory allocated for results.

udm_get_doc_count (PHP 4 CVS only)

Get total number of documents in database.

```
int udm_get_doc_count (int agent)
```

udm_get_doc_count() returns nuimber of documents in database.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

> **Poznámka:** This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field (PHP 4 CVS only)

Fetch mnoGoSearch result field

```
string udm_get_res_field (int res, int row, int field)
```

udm_get_res_field() returns result field value on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).

- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param (PHP 4 CVS only)

Get mnoGoSearch result parameters

```
string udm_get_res_param (int res, int param)
```

udm_get_res_param() returns result parameter value on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_load_ispell_data (PHP 4 CVS only)

Load ispell data

```
int udm_load_ispell_data (int agent, int var, string val1, string val2, int flag)
```

udm_load_ispell_data() loads ispell data. Returns TRUE on success, FALSE on error.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

var - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use **udm_free_ispell_data()**, even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is true for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *val1* and *val2* are ignored and should be left blank. *flag* should be equal to 1.

  Poznámka: *flag* indicates that after loading ispell data from defined source it sould be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every call, but only after the last one. Since in

db mode all the data is loaded by one call, this parameter should have the value `1`. In this mode in case of error, e.g. if ispell tables are absent, the function will return `FALSE` and code and error message will be accessible through **udm_error()** and **udm_errno()**.

Example:

```
if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_DB,”,”,1)) {
   printf("Error #%d: ’%s’\n",Udm_Errno($udm),Udm_Error($udm));
   exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’ru’,’/opt/ispell/ru.aff’,0
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’en’,’/opt/ispell/en.dict’,
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’ru’,’/opt/ispell/ru.dict’,
   exit;
}
```

  Poznámka: *flag* is equal to `1` only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’ru’,’/opt/ispell/ru.aff’,0
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’en’,’/opt/ispell/en.dict’,
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’ru’,’/opt/ispell/ru.dict’,
   exit;
}
```

  Poznámka: *flag* is equal to `1` only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client

(search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by **udm_find()** function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

> **Poznámka:** This function is available in mnoGoSearch 3.1.12 or later.

Example:

```
    if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SERVER,”,”,1)) {
     printf("Error loading ispell data from server<br>\n");
     exit;
}
```

udm_set_agent_param (PHP 4 CVS only)

Set mnoGoSearch agent session parameters

```
int udm_set_agent_param (int agent, int var, string val)
```

udm_set_agent_param() returns `TRUE` on success, `FALSE` on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See **udm_find()** for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and buuilt-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

  Examples of phrase search:

  "Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...

- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let’s imagine that we have these weights:

  URLWeight 1

  BodyWeight 2

  TitleWeight 4

  KeywordWeight 8

  DescWeight 16

  As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

  This parameter’s value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

  E is a factor for weight 1 (URL Weight bit)

  D is a factor for weight 2 (BodyWeight bit)

  C is a factor for weight 4 (TitleWeight bit)

  B is a factor for weight 8 (KeywordWeight bit)

  A is a factor for weight 16 (DescWeight bit)

  Examples:

  UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

  UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

  UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Desctription but not through URL and Body.

  UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

  Description with factor 15 (F hex)

  Keywords with factor 9

  Title with factor 4

  Body with factor 2

  URL with factor 1

  If UDM_PARAM_WEIGHT_FACTOR variable is ommited, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

  UDM_MATCH_BEGIN - word beginning match;

  UDM_MATCH_END - word ending match;

  UDM_MATCH_WORD - whole word match;

  UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this paraneter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.
- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site’s spelling checking purposes. In order to enable ispell, you have to load ispell data with **udm_load_ispell_data()**.
- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The corsswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.

  Poznámka: Crosswords are supported only in mnoGoSearch 3.1.11 or later.

XLVII. mSQL functions

These functions allow you to access mSQL database servers. In order to have these functions available, you must compile php with msql support by using the `-with-msql[=dir]` option. The default location is /usr/local/Hughes.

More information about mSQL can be found at http://www.hughes.com.au/.

msql (PHP 3, PHP 4)

Send mSQL query

```
int msql (string database, string query, int link_identifier)
```

Returns a positive mSQL query identifier to the query result, or false on error.

msql() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the mSQL server and if no such link is found it'll try to create one as if **msql_connect()** was called with no arguments (see **msql_connect()**).

msql_affected_rows (PHP 3>= 3.0.6, PHP 4)

Returns number of affected rows

```
int msql_affected_rows (int query_identifier)
```

Returns number of affected ("touched") rows by a specific query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

See also: **msql_query()**.

msql_close (PHP 3, PHP 4)

Close mSQL connection

```
int msql_close (int link_identifier)
```

Returns true on success, false on error.

msql_close() closes the link to a mSQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

msql_close() will not close persistent links generated by **msql_pconnect()**.

See also: **msql_connect()** and **msql_pconnect()**.

msql_connect (PHP 3, PHP 4)

Open mSQL connection

```
int msql_connect ([string hostname [, string hostname[:port] [, string username [,
string password]]]])
```

Returns a positive mSQL link identifier on success, or false on error.

msql_connect() establishes a connection to a mSQL server. The hostname argument is optional, and if it's missing, localhost is assumed.

In case a second call is made to **msql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **msql_close()**.

See also **msql_pconnect()**, **msql_close()**.

msql_create_db (PHP 3, PHP 4)

Create mSQL database

```
int msql_create_db (string database name [, int link_identifier])
```

msql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: **msql_drop_db()**.

msql_createdb (PHP 3, PHP 4)

Create mSQL database

```
int msql_createdb (string database name [, int link_identifier])
```

Identical to **msql_create_db()**.

msql_data_seek (PHP 3, PHP 4)

Move internal row pointer

```
int msql_data_seek (int query_identifier, int row_number)
```

Returns true on success, false on failure.

msql_data_seek() moves the internal row pointer of the mSQL result associated with the specified query identifier to pointer to the specifyed row number. The next call to **msql_fetch_row()** would return that row.

See also: **msql_fetch_row()**.

msql_dbname (PHP 3, PHP 4)

Get current mSQL database name

```
string msql_dbname (int query_identifier, int i)
```

msql_dbname() returns the database name stored in position *i* of the result pointer returned from the **msql_listdbs()** function. The **msql_numrows()** function can be used to determine how many database names are available.

msql_drop_db (PHP 3, PHP 4)

Drop (delete) mSQL database

```
int msql_drop_db (string database_name, int link_identifier)
```

Returns true on success, false on failure.

msql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: **msql_create_db()**.

msql_dropdb (PHP 3, PHP 4)

Drop (delete) mSQL database

See **msql_drop_db()**.

msql_error (PHP 3, PHP 4)

Returns error message of last msql call

```
string msql_error ()
```

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

msql_fetch_array (PHP 3, PHP 4)

Fetch row as array

```
int msql_fetch_array (int query_identifier [, int result_type])
```

Returns an array that corresponds to the fetched row, or false if there are no more rows.

msql_fetch_array() is an extended version of **msql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

The second optional argument *result_type* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MYSQL_BOTH.

Be careful if you are retrieving results from a query that may return a record that contains only one field that has a value of 0 (or an empty string, or NULL).

An important thing to note is that using **msql_fetch_array()** is NOT significantly slower than using **msql_fetch_row()**, while it provides a significant added value.

For further details, also see **msql_fetch_row()**.

msql_fetch_field (PHP 3, PHP 4)

Get field information

```
object msql_fetch_field (int query_identifier, int field_offset)
```

Returns an object containing field information

msql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **msql_fetch_field()** is retreived.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- not_null - 1 if the column cannot be null
- primary_key - 1 if the column is a primary key
- unique - 1 if the column is a unique key
- type - the type of the column

See also **msql_field_seek()**.

msql_fetch_object (PHP 3, PHP 4)

Fetch row as object

```
int msql_fetch_object (int query_identifier [, int result_type])
```

Returns an object with properties that correspond to the fetched row, or false if there are no more rows.

msql_fetch_object() is similar to **msql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional second argument *result_type* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Speed-wise, the function is identical to **msql_fetch_array()**, and almost as quick as **msql_fetch_row()** (the difference is insignificant).

See also: **msql_fetch_array()** and **msql_fetch_row()**.

msql_fetch_row (PHP 3, PHP 4)

Get row as enumerated array

```
array msql_fetch_row (int query_identifier)
```

Returns an array that corresponds to the fetched row, or false if there are no more rows.

msql_fetch_row() fetches one row of data from the result associated with the specified query identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **msql_fetch_row()** would return the next row in the result set, or false if there are no more rows.

See also: **msql_fetch_array()**, **msql_fetch_object()**, **msql_data_seek()**, and **msql_result()**.

msql_fieldname (PHP 3, PHP 4)

Get field name

```
string msql_fieldname (int query_identifier, int field)
```

msql_fieldname() returns the name of the specified field. *query_identifier* is the query identifier, and *field* is the field index. `msql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

msql_field_seek (PHP 3, PHP 4)

Set field offset

```
int msql_field_seek (int query_identifier, int field_offset)
```

Seeks to the specified field offset. If the next call to **msql_fetch_field()** won't include a field offset, this field would be returned.

See also: **msql_fetch_field()**.

msql_fieldtable (PHP 3, PHP 4)

Get table name for field

```
int msql_fieldtable (int query_identifier, int field)
```

Returns the name of the table *field* was fetched from.

msql_fieldtype (PHP 3, PHP 4)

Get field type

```
string msql_fieldtype (int query_identifier, int i)
```

msql_fieldtype() is similar to the **msql_fieldname()** function. The arguments are identical, but the field type is returned. This will be one of "int", "char" or "real".

msql_fieldflags (PHP 3, PHP 4)

Get field flags

```
string msql_fieldflags (int query_identifier, int i)
```

msql_fieldflags() returns the field flags of the specified field. Currently this is either, "not null", "primary key", a combination of the two or "" (an empty string).

msql_fieldlen (PHP 3, PHP 4)

Get field length

```
int msql_fieldlen (int query_identifier, int i)
```

msql_fieldlen() returns the length of the specified field.

msql_free_result (PHP 3, PHP 4)

Free result memory

```
int msql_free_result (int query_identifier)
```

msql_free_result() frees the memory associated with *`query_identifier`*. When PHP completes a request, this memory is freed automatically, so you only need to call this function when you want to make sure you don't use too much memory while the script is running.

msql_freeresult (PHP 3, PHP 4)

Free result memory

See **msql_free_result()**

msql_list_fields (PHP 3, PHP 4)

List result fields

```
int msql_list_fields (string database, string tablename)
```

msql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **msql_fieldflags()**, **msql_fieldlen()**, **msql_fieldname()**, and **msql_fieldtype()**. A query identifier is a positive integer. The function returns `-1` if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@msql_list_fields()` then this error string will also be printed out.

See also **msql_error()**.

msql_listfields (PHP 3, PHP 4)

List result fields

See **msql_list_fields()**.

msql_list_dbs (PHP 3, PHP 4)

List mSQL databases on server

```
int msql_list_dbs(void);
```

msql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the **msql_dbname()** function to traverse this result pointer.

msql_listdbs (PHP 3, PHP 4)

List mSQL databases on server

See **msql_list_dbs()**.

msql_list_tables (PHP 3, PHP 4)

List tables in an mSQL database

```
int msql_list_tables (string database)
```

msql_list_tables() takes a database name and result pointer much like the **msql()** function. The **msql_tablename()** function should be used to extract the actual table names from the result pointer.

msql_listtables (PHP 3, PHP 4)

List tables in an mSQL database

See **msql_list_tables()**.

msql_num_fields (PHP 3, PHP 4)

Get number of fields in result

```
int msql_num_fields (int query_identifier)
```

msql_num_fields() returns the number of fields in a result set.

See also: **msql()**, **msql_query()**, **msql_fetch_field()**, and **msql_num_rows()**.

msql_num_rows (PHP 3, PHP 4)

Get number of rows in result

```
int msql_num_rows (int query_identifier)
```

Msql_num_rows() returns the number of rows in a result set.

See also: **msql()**, **msql_query()**, and **msql_fetch_row()**.

msql_numfields (PHP 3, PHP 4)

Get number of fields in result

```
int msql_numfields (int query_identifier)
```

Identical to **msql_num_fields()**.

msql_numrows (PHP 3, PHP 4)

Get number of rows in result

```
int msql_numrows(void);
```

Identical to **msql_num_rows()**.

msql_pconnect (PHP 3, PHP 4)

Open persistent mSQL connection

```
int msql_pconnect ([string hostname [, string hostname[:port] [, string username [,
string password]]]])
```

Returns a positive mSQL persistent link identifier on success, or false on error.

msql_pconnect() acts very much like **msql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**msql_close()** will not close links established by **msql_pconnect()**).

This type of links is therefore called 'persistent'.

msql_query (PHP 3, PHP 4)

Send mSQL query

```
int msql_query (string query, int link_identifier)
```

msql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **msql_connect()** was called, and use it.

Returns a positive mSQL query identifier on success, or false on error.

See also: **msql()**, **msql_select_db()**, and **msql_connect()**.

msql_regcase (PHP 3, PHP 4)

Make regular expression for case insensitive match

See **sql_regcase()**.

msql_result (PHP 3, PHP 4)

Get result data

```
int msql_result (int query_identifier, int i, mixed field)
```

Returns the contents of the cell at the row and offset in the specified mSQL result set.

msql_result() returns the contents of one cell from a mSQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from ...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **msql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **msql_fetch_row()**, **msql_fetch_array()**, and **msql_fetch_object()**.

msql_select_db (PHP 3, PHP 4)

Select mSQL database

```
int msql_select_db (string database_name, int link_identifier)
```

Returns true on success, false on error.

msql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **msql_connect()** was called, and use it.

Every subsequent call to **msql_query()** will be made on the active database.

See also: **msql_connect()**, **msql_pconnect()**, and **msql_query()**.

msql_selectdb (PHP 3, PHP 4)

Select mSQL database

See **msql_select_db()**.

msql_tablename (PHP 3, PHP 4)

Get table name of field

```
string msql_tablename (int query_identifier, int field)
```

msql_tablename() takes a result pointer returned by the **msql_list_tables()** function as well as an integer index and returns the name of a table. The **msql_numrows()** function may be used to determine the number of tables in the result pointer.

Příklad 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XLVIII. MySQL functions

These functions allow you to access MySQL database servers. In order to have these functions available, you must compile php with mysql support by using the `-with-mysql` option. If you use this option without specifying the path to mysql, php will use the built-in mysql client libraries. Users who run other applications that use mysql (for example, running php3 and php4 as concurrent apache modules, or auth-mysql) should always specify the path to mysql: `-with-mysql=/path/to/mysql`. This will force php to use the client libraries installed by mysql, avoiding any conflicts.

More information about MySQL can be found at http://www.mysql.com/.

Documentation for MySQL can be found at http://www.mysql.com/documentation/.

mysql_affected_rows (PHP 3, PHP 4)

Get number of affected rows in previous MySQL operation

```
int mysql_affected_rows ([int link_identifier])
```

mysql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by **mysql_connect()** is assumed.

> **Poznámka:** If you are using transactions, you need to call **mysql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Poznámka:** When using UPDATE, MySQL will not update columns where the new value is the same as the old value. This creates the possiblity that **mysql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

mysql_affected_rows() does not work with SELECT statements; only on statements which modify records. To retrieve the number of rows returned by a SELECT, use **mysql_num_rows()**.

If the last query failed, this function will return -1.

See also: **mysql_num_rows()**.

mysql_change_user (PHP 3>= 3.0.13)

Change logged in user of the active connection

```
int mysql_change_user (string user, string password [, string database [, int
link_identifier]])
```

mysql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

> **Poznámka:** This function was introduced in PHP 3.0.13 and requires MySQL 3.23.3 or higher.

mysql_close (PHP 3, PHP 4)

Close MySQL connection

```
int mysql_close ([int link_identifier])
```

Returns: true on success, false on error.

mysql_close() closes the connection to the MySQL server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **mysql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Poznámka: mysql_close() will not close persistent links created by **mysql_pconnect()**.

Příklad 1. MySQL close example

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);
?>
```

See also: **mysql_connect()**, and **mysql_pconnect()**.

mysql_connect (PHP 3, PHP 4)

Open a connection to a MySQL Server

```
int mysql_connect ([string hostname [:port] [:/path/to/socket] [, string username [,
string password]]])
```

Returns a positive MySQL link identifier on success, or an error message on failure.

mysql_connect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *host:port* = 'localhost:3306', *username* = name of the user that owns the server process and *password* = empty password.

The hostname string can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

Poznámka: Support for ":port" was added in PHP 3.0B4.

Support for ":/path/to/socket" was added in PHP 3.0.10.

You can suppress the error message on failure by prepending '@' to the function name.

If a second call is made to **mysql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mysql_close()**.

Příklad 1. MySQL connect example

```
<?php

    $link = mysql_connect ("localhost", "username", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);

?>
```

See also **mysql_pconnect()**, and **mysql_close()**.

mysql_create_db (PHP 3, PHP 4)

Create a MySQL database

```
int mysql_create_db (string database name [, int link_identifier])
```

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Příklad 1. MySQL create database example

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("Could not connect");
    if (mysql_create_db ("my_db")) {
        print ("Database created successfully\n");
    } else {
        printf ("Error creating database: %s\n", mysql_error ());
    }
?>
```

For downwards compatibility **mysql_createdb()** can also be used.

See also: **mysql_drop_db()**.

mysql_data_seek (PHP 3, PHP 4)

Move internal result pointer

```
int mysql_data_seek (int result_identifier, int row_number)
```

Returns: true on success, false on failure.

mysql_data_seek() moves the internal row pointer of the MySQL result associated with the specified result identifier to point to the specified row number. The next call to **mysql_fetch_row()** would return that row.

Row_number starts at 0.

Příklad 1. MySQL data seek example

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("Could not connect");

    mysql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i-) {
        if (!mysql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = mysql_fetch_object ($result)))
            continue;

        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    mysql_free_result ($result);
?>
```

mysql_db_name (PHP 3>= 3.0.6, PHP 4)

Get result data

```
int mysql_db_name (int result, int row [, mixed field])
```

mysql_db_name() takes as its first parameter the result pointer from a call to **mysql_list_dbs()**. The *row* parameter is an index into the result set.

If an error occurs, FALSE is returned. Use **mysql_errno()** and **mysql_error()** to determine the nature of the error.

Příklad 1. Mysql_db_name() example

```
<?php
error_reporting(E_ALL);

mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs();

$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
?>
```

For backward compatibility, **mysql_dbname()** is also accepted. This is deprecated, however.

mysql_db_query (PHP 3, PHP 4)

Send a MySQL query

```
int mysql_db_query (string database, string query [, int link_identifier])
```

Returns: A positive MySQL result identifier to the query result, or false on error.

mysql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the MySQL server and if no such link is found it'll try to create one as if **mysql_connect()** was called with no arguments

See also **mysql_connect()**.

For downwards compatibility **mysql()** can also be used.

mysql_drop_db (PHP 3, PHP 4)

Drop (delete) a MySQL database

```
int mysql_drop_db (string database_name [, int link_identifier])
```

Returns: true on success, false on failure.

mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: **mysql_create_db()**. For downward compatibility **mysql_dropdb()** can also be used.

mysql_errno (PHP 3, PHP 4)

Returns the numerical value of the error message from previous MySQL operation

```
int mysql_errno ([int link_identifier])
```

Returns the error number from the last mySQL function, or `0` (zero) if no error occurred.

Errors coming back from the mySQL database backend no longer issue warnings. Instead, use **mysql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed mySQL function (not including **mysql_error()** and **mysql_errno()**), so if you want to use it, make sure you check the value before calling another mySQL function.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

See also: **mysql_error()**

mysql_error (PHP 3, PHP 4)

Returns the text of the error message from previous MySQL operation

```
string mysql_error ([int link_identifier])
```

Returns the error text from the last mySQL function, or `''` (the empty string) if no error occurred.

Errors coming back from the mySQL database backend no longer issue warnings. Instead, use **mysql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed mySQL function (not including **mysql_error()** and **mysql_errno()**), so if you want to use it, make sure you check the value before calling another mySQL function.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

See also: **mysql_errno()**

mysql_fetch_array (PHP 3, PHP 4)

Fetch a result row as an associative array, a numeric array, or both.

```
array mysql_fetch_array (int result [, int result_type])
```

Returns an array that corresponds to the fetched row, or false if there are no more rows.

mysql_fetch_array() is an extended version of **mysql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **mysql_fetch_array()** is NOT significantly slower than using **mysql_fetch_row()**, while it provides a significant added value.

The optional second argument *result_type* in **mysql_fetch_array()** is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_BOTH. (This feature was added in PHP 3.0.7)

For further details, see also **mysql_fetch_row()** and **mysql_fetch_assoc()**.

Příklad 1. Mysql_fetch_array()

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database","select user_id, fullname from table");
while ($row = mysql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
mysql_free_result ($result);
?>
```

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Fetch a result row as an associative array

```
array mysql_fetch_assoc (int result)
```

Returns an associative array that corresponds to the fetched row, or false if there are no more rows.

mysql_fetch_assoc() is equivalent to calling **mysql_fetch_array()** with MYSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way **mysql_fetch_array()** originally worked. If you need the numeric indices as well as the associative, use **mysql_fetch_array()**.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use **mysql_fetch_array()** and have it return the numeric indices as well.

An important thing to note is that using **mysql_fetch_assoc()** is NOT significantly slower than using **mysql_fetch_row()**, while it provides a significant added value.

For further details, see also **mysql_fetch_row()** and **mysql_fetch_array()**.

Příklad 1. Mysql_fetch_assoc()

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database","select * from table");
while ($row = mysql_fetch_assoc ($result)) {
    echo $row["user_id"];
```

```
    echo $row["fullname"];
}
mysql_free_result ($result);
?>
```

mysql_fetch_field (PHP 3, PHP 4)

Get column information from a result and return as an object

```
object mysql_fetch_field (int result [, int field_offset])
```

Returns an object containing field information.

mysql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mysql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be null
- primary_key - 1 if the column is a primary key
- unique_key - 1 if the column is a unique key
- multiple_key - 1 if the column is a non-unique key
- numeric - 1 if the column is numeric
- blob - 1 if the column is a BLOB
- type - the type of the column
- unsigned - 1 if the column is unsigned
- zerofill - 1 if the column is zero-filled

Příklad 1. Mysql_fetch_field()

```
<?php
mysql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = mysql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < mysql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = mysql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
```

```
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</PRE>";
    $i++;
}
mysql_free_result ($result);
?>
```

See also **mysql_field_seek()**.

mysql_fetch_lengths (PHP 3, PHP 4)

Get the length of each output in a result

```
array mysql_fetch_lengths (int result)
```

Returns: An array that corresponds to the lengths of each field in the last row fetched by **mysql_fetch_row()**, or false on error.

mysql_fetch_lengths() stores the lengths of each result column in the last row returned by **mysql_fetch_row()**, **mysql_fetch_array()**, and **mysql_fetch_object()** in an array, starting at offset 0.

See also: **mysql_fetch_row()**.

mysql_fetch_object (PHP 3, PHP 4)

Fetch a result row as an object

```
object mysql_fetch_object (int result [, int result_type])
```

Returns an object with properties that correspond to the fetched row, or false if there are no more rows.

mysql_fetch_object() is similar to **mysql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_BOTH.

Speed-wise, the function is identical to **mysql_fetch_array()**, and almost as quick as **mysql_fetch_row()** (the difference is insignificant).

Příklad 1. mysql_fetch_object() example

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database", "select * from table");
while ($row = mysql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result ($result);
?>
```

See also: **mysql_fetch_array()** and **mysql_fetch_row()**.

mysql_fetch_row (PHP 3, PHP 4)

Get a result row as an enumerated array

```
array mysql_fetch_row (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mysql_fetch_row()** would return the next row in the result set, or false if there are no more rows.

See also: **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()**, and **mysql_result()**.

mysql_field_flags (PHP 3, PHP 4)

Get the flags associated with the specified field in a result

```
string mysql_field_flags (int result, int field_offset)
```

mysql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using **explode()**.

The following flags are reported, if your version of MySQL is current enough to support them: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

For downward compatibility **mysql_fieldflags()** can also be used.

mysql_field_name (PHP 3, PHP 4)

Get the name of the specified field in a result

```
string mysql_field_name (int result, int field_index)
```

mysql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Poznámka:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Příklad 1. mysql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = mysql_db_query("users", "select * from users", $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

For downwards compatibility **mysql_fieldname()** can also be used.

mysql_field_len (PHP 3, PHP 4)

Returns the length of the specified field

```
int mysql_field_len (int result, int field_offset)
```

mysql_field_len() returns the length of the specified field.

For downward compatibility **mysql_fieldlen()** can also be used.

mysql_field_seek (PHP 3, PHP 4)

Set result pointer to a specified field offset

```
int mysql_field_seek (int result, int field_offset)
```

Seeks to the specified field offset. If the next call to **mysql_fetch_field()** doesn't include a field offset, the field offset specified in **mysql_field_seek()** will be returned.

See also: **mysql_fetch_field()**.

mysql_field_table (PHP 3, PHP 4)

Get name of the table the specified field is in

```
string mysql_field_table (int result, int field_offset)
```

Returns the name of the table that the specifed field is in.

For downward compatibility **mysql_fieldtable()** can also be used.

mysql_field_type (PHP 3, PHP 4)

Get the type of the specified field in a result

```
string mysql_field_type (int result, int field_offset)
```

mysql_field_type() is similar to the **mysql_field_name()** function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the MySQL documentation (http://www.mysql.com/documentation/).

Příklad 1. mysql field types

```
<?php
```

```
mysql_connect ("localhost:3306");
mysql_select_db ("wisconsin");
$result = mysql_query ("SELECT * FROM onek");
$fields = mysql_num_fields ($result);
$rows   = mysql_num_rows ($result);
$i = 0;
$table = mysql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = mysql_field_type  ($result, $i);
    $name  = mysql_field_name  ($result, $i);
    $len   = mysql_field_len   ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
mysql_close();

?>
```

For downward compatibility **mysql_fieldtype()** can also be used.

mysql_free_result (PHP 3, PHP 4)

Free result memory

```
int mysql_free_result (int result)
```

mysql_free_result() will free all memory associated with the result identifier *result*.

mysql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

For downward compatibility **mysql_freeresult()** can also be used.

mysql_insert_id (PHP 3, PHP 4)

Get the id generated from the previous INSERT operation

```
int mysql_insert_id ([int link_identifier])
```

mysql_insert_id() returns the ID generated for an AUTO_INCREMENT column by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

mysql_insert_id() returns 0 if the previous query does not generate an AUTO_INCREMENT value. If you need to save the value for later, be sure to call mysql_insert_id() immediately after the query that generates the value.

> **Poznámka:** The value of the MySQL SQL function `LAST_INSERT_ID()` always contains the most recently generated AUTO_INCREMENT value, and is not reset between queries.

> **Varování**
>
> **mysql_insert_id()** converts the return type of the native MySQL C API function `mysql_insert_id()` to a type of `long`. If your AUTO_INCREMENT column has a column type of BIGINT, the value returned by **mysql_insert_id()** will be incorrect. Instead, use the internal MySQL SQL function `LAST_INSERT_ID()`.

mysql_list_dbs (PHP 3, PHP 4)

List databases available on a MySQL server

```
int mysql_list_dbs ([int link_identifier])
```

mysql_list_dbs() will return a result pointer containing the databases available from the current mysql daemon. Use the **mysql_tablename()** function to traverse this result pointer.

Příklad 1. mysql_list_dbs() example

```
$link = mysql_connect('localhost', 'myname', 'secret');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
  echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

Poznámka: The above code would just as easily work with **mysql_fetch_row()** or other similar functions.

For downward compatibility **mysql_listdbs()** can also be used.

mysql_list_fields (PHP 3, PHP 4)

List MySQL result fields

```
int mysql_list_fields (string database_name, string table_name [, int link_identifier])
```

mysql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()**, and **mysql_field_type()**.

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@mysql()` then this error string will also be printed out.

Příklad 1. mysql_list_fields() example

```
$link = mysql_connect('localhost', 'myname', 'secret');

$fields = mysql_list_fields("database1", "table1", $link);
$columns = mysql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
  echo mysql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

For downward compatibility **mysql_listfields()** can also be used.

mysql_list_tables (PHP 3, PHP 4)

List tables in a MySQL database

```
int mysql_list_tables (string database [, int link_identifier])
```

mysql_list_tables() takes a database name and returns a result pointer much like the **mysql_db_query()** function. The **mysql_tablename()** function should be used to extract the actual table names from the result pointer.

For downward compatibility **mysql_listtables()** can also be used.

mysql_num_fields (PHP 3, PHP 4)

Get number of fields in result

```
int mysql_num_fields (int result)
```

mysql_num_fields() returns the number of fields in a result set.

See also: **mysql_db_query()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

For downward compatibility **mysql_numfields()** can also be used.

mysql_num_rows (PHP 3, PHP 4)

Get number of rows in result

```
int mysql_num_rows (int result)
```

mysql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use **mysql_affected_rows()**.

Příklad 1. mysql_num_rows() example

```
<?php

$link = mysql_connect("localhost", "username", "password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: **mysql_affected_rows()**, **mysql_connect()**, **mysql_select_db()** and **mysql_query()**.

For downward compatibility **mysql_numrows()** can also be used.

mysql_pconnect (PHP 3, PHP 4)

Open a persistent connection to a MySQL Server

```
int mysql_pconnect ([string hostname [:port] [:/path/to/socket] [, string username [, string password]]])
```

Returns: A positive MySQL persistent link identifier on success, or false on error.

mysql_pconnect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *host:port* = 'localhost:3306', *username* = name of the user that owns the server process and *password* = empty password.

The hostname string can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

Poznámka: Support for ":port" wass added in 3.0B4.

Support for the ":/path/to/socket" was added in 3.0.10.

mysql_pconnect() acts very much like **mysql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**mysql_close()** will not close links established by **mysql_pconnect()**).

This type of links is therefore called 'persistent'.

mysql_query (PHP 3, PHP 4)

Send a MySQL query

```
int mysql_query (string query [, int link_identifier])
```

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **mysql_connect()** was called with no arguments, and use it.

Poznámka: The query string should not end with a semicolon.

mysql_query() returns TRUE (non-zero) or FALSE to indicate whether or not the query succeeded. A return value of TRUE means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **mysql_query()** fails and returns FALSE:

Příklad 1. mysql_query()

```
<?php
```

```
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **mysql_query()** fails and returns FALSE:

Příklad 2. mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() will also fail and return FALSE if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call **mysql_num_rows()** to find out how many rows were returned for a SELECT statment or **mysql_affected_rows()** to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **mysql_query()** returns a new result identifier that you can pass to **mysql_result()**. When you are done with the result set, you can free the resources associated with it by calling **mysql_free_result()**. Although, the memory will automatically be freed at the end of the script's execution.

See also: **mysql_affected_rows()**, **mysql_db_query()**, **mysql_free_result()**, **mysql_result()**, **mysql_select_db()**, and **mysql_connect()**.

mysql_result (PHP 3, PHP 4)

Get result data

```
mixed mysql_result (int result, int row [, mixed field])
```

mysql_result() returns the contents of one cell from a MySQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mysql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **mysql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: **mysql_fetch_row()**, **mysql_fetch_array()**, and **mysql_fetch_object()**.

mysql_select_db (PHP 3, PHP 4)

Select a MySQL database

```
int mysql_select_db (string database_name [, int link_identifier])
```

Returns: true on success, false on error.

mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mysql_connect()** was called, and use it.

Every subsequent call to **mysql_query()** will be made on the active database.

See also: **mysql_connect()**, **mysql_pconnect()**, and **mysql_query()**.

For downward compatibility **mysql_selectdb()** can also be used.

mysql_tablename (PHP 3, PHP 4)

Get table name of field

```
string mysql_tablename (int result, int i)
```

mysql_tablename() takes a result pointer returned by the **mysql_list_tables()** function as well as an integer index and returns the name of a table. The **mysql_num_rows()** function may be used to determine the number of tables in the result pointer.

Příklad 1. Mysql_tablename() Example

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XLIX. Network Functions

checkdnsrr (PHP 3, PHP 4)

Check DNS records corresponding to a given Internet host name or IP address

```
int checkdnsrr (string host [, string type])
```

Searches DNS for records of type *type* corresponding to *host*. Returns true if any records are found; returns false if no records were found or if an error occurred.

type may be any one of: A, MX, NS, SOA, PTR, CNAME, or ANY. The default is MX.

Host may either be the IP address in dotted-quad notation or the host name.

See also **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()**, and the named(8) manual page.

closelog (PHP 3, PHP 4)

Close connection to system logger

```
int closelog(void);
```

Closelog() closes the descriptor being used to write to the system logger. The use of **closelog()** is optional.

See also **define_syslog_variables()**, **syslog()** and **openlog()**.

debugger_off (PHP 3)

Disable internal PHP debugger

```
int debugger_off(void);
```

Disables the internal PHP debugger. The debugger is still under development.

debugger_on (PHP 3)

Enable internal PHP debugger

```
int debugger_on (string address)
```

Enables the internal PHP debugger, connecting it to *address*. The debugger is still under development.

define_syslog_variables (PHP 3, PHP 4)

Initializes all syslog related constants

```
void define_syslog_varaibles (void)
```

Initializes all constants used in the syslog functions.

See also **openlog()**, **syslog()** and **closelog()**.

fsockopen (PHP 3, PHP 4)

Open Internet or Unix domain socket connection

```
int fsockopen (string [udp://]hostname, int port [, int errno [, string errstr [,
double timeout]]])
```

Initiates a stream connection in the Internet (AF_INET, using TCP or UDP) or Unix (AF_UNIX) domain. For the Internet domain, it will open a TCP socket connection to *hostname* on port *port*. *hostname* may in this case be either a fully qualified domain name or an IP address. For UDP connections, you need to explicitly specify the protocol: *udp://hostname*. For the Unix domain, *hostname* will be used as the path to the socket, *port* must be set to 0 in this case. The optional *timeout* can be used to set a timeout in seconds for the connect system call.

Fsockopen() returns a file pointer which may be used together with the other file functions (such as **fgets()**, **fgetss()**, **fputs()**, **fclose()**, and **feof()**).

If the call fails, it will return false and if the optional *errno* and *errstr* arguments are present they will be set to indicate the actual system level error that occurred on the system-level `connect()` call. If the returned errno is 0 and the function returned false, it is an indication that the error occurred before the `connect()` call. This is most likely due to a problem initializing the socket. Note that the *errno* and *errstr* arguments must be passed by reference.

Depending on the environment, the Unix domain or the optional connect timeout may not be available.

The socket will by default be opened in blocking mode. You can switch it to non-blocking mode by using **socket_set_blocking()**.

Příklad 1. Fsockopen() Example

```
$fp = fsockopen ("www.php.net", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br>\n";
} else {
    fputs ($fp, "GET / HTTP/1.0\r\n\r\n");
    while (!feof($fp)) {
        echo fgets ($fp,128);
    }
    fclose ($fp);
}
```

The example below shows how to retrieve the day and time from the UDP service "daytime" (port 13) in your own machine.

Příklad 2. Using UDP connection

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
    echo "ERROR: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

Poznámka: The timeout parameter was introduced in PHP 3.0.9 and UDP support was added in PHP 4.

See also: **pfsockopen()**, **socket_set_blocking()**, **socket_set_timeout()**, **fgets()**, **fgetss()**, **fputs()**, **fclose()**, and **feof()**).

gethostbyaddr (PHP 3, PHP 4)

Get the Internet host name corresponding to a given IP address

```
string gethostbyaddr (string ip_address)
```

Returns the host name of the Internet host specified by *ip_address*. If an error occurs, returns *ip_address*.

See also **gethostbyname()**.

gethostbyname (PHP 3, PHP 4)

Get the IP address corresponding to a given Internet host name

```
string gethostbyname (string hostname)
```

Returns the IP address of the Internet host specified by *hostname*.

See also **gethostbyaddr()**.

gethostbynamel (PHP 3, PHP 4)

Get a list of IP addresses corresponding to a given Internet host name

```
array gethostbynamel (string hostname)
```

Returns a list of IP addresses to which the Internet host specified by *hostname* resolves.

See also **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()**, and the `named(8)` manual page.

getmxrr (PHP 3, PHP 4)

Get MX records corresponding to a given Internet host name

```
int getmxrr (string hostname, array mxhosts [, array weight])
```

Searches DNS for MX records corresponding to *hostname*. Returns true if any records are found; returns false if no records were found or if an error occurred.

A list of the MX records found is placed into the array *mxhosts*. If the *weight* array is given, it will be filled with the weight information gathered.

See also **checkdnsrr()**, **gethostbyname()**, **gethostbynamel()**, **gethostbyaddr()**, and the `named(8)` manual page.

getprotobyname (PHP 4 >= 4.0b4)

Get protocol number associated with protocol name

```
int getprotobyname (string name)
```

Getprotobyname() returns the protocol number associated with the protocol *name* as per /etc/protocols.

See also: **getprotobynumber()**.

getprotobynumber (PHP 4 >= 4.0b4)

Get protocol name associated with protocol number

```
string getprotobynumber (int number)
```

Getprotobynumber() returns the protocol name associated with protocol *number* as per /etc/protocols.

See also: **getprotobyname()**.

getservbyname (PHP 4 >= 4.0b4)

Get port number associated with an Internet service and protocol

```
int getservbyname (string service, string protocol)
```

Getservbyname() returns the Internet port which corresponds to *service* for the specified *protocol* as per /etc/services. *protocol* is either TCP or UDP.

See also: **getservbyport()**.

getservbyport (PHP 4 >= 4.0b4)

Get Internet service which corresponds to port and protocol

```
string getservbyport (int port, string protocol)
```

Getservbyport() returns the Internet service associated with *port* for the specified *protocol* as per /etc/services. *protocol* is either TCP or UDP.

See also: **getservbyname()**.

ip2long (PHP 4 >= 4.0RC1)

Converts a string containing an (IPv4) Internet Protocol dotted address into a proper address.

```
int ip2long (string ip_address)
```

The function **ip2long()** generates an IPv4 Internet network address from its Internet standard format (dotted string) representation.

Příklad 1. Ip2long() Example

```
<?php
$ip = gethostbyname("www.php.net");
$out = "The following URLs are equivalent:<br>\n";
$out .= "http://www.php.net/, http://".$ip."/, and http://".ip2long($ip)."/<br>\n";
echo $out;
?>
```

See also: **long2ip()**

long2ip (PHP 4 >= 4.0RC1)

Converts an (IPv4) Internet network address into a string in Internet standard dotted format

```
string long2ip (int proper_address)
```

The function **long2ip()** generates an Internet address in dotted format (i.e.: aaa.bbb.ccc.ddd) from the proper address representation.

See also: **ip2long()**

openlog (PHP 3, PHP 4)

Open connection to system logger

```
int openlog (string ident, int option, int facility)
```

Openlog() opens a connection to the system logger for a program. The string *ident* is added to each message. Values for *option* and *facility* are given below. The *option* argument is used to indicate what loggin options will be used when generating a log message. The *facility* argument is used to specify what type of program is logging the message. This allows you to specify (in your machine's syslog configuration) how messages coming from different facilities will be handled. The use of **openlog()** is optional. It will automatically be called by **syslog()** if necessary, in which case *ident* will default to false.

Tabulka 1. Openlog() Options

Constant	Description
LOG_CONS	if there is an error while sending data to the system logger, write directly to the system console
LOG_NDELAY	open the connection to the logger immediately
LOG_ODELAY	(default) delay openning the connection until the first message is logged
LOG_PERROR	print log message also to standard error
LOG_PID	include PID with each message

You can use one or more of this options. When using multiple options you need to `OR` them, i.e. to open the connection immediately, write to the consoloe and include the PID in each message, you will use: `LOG_CONS | LOG_NDELAY | LOG_PID`

Tabulka 2. Openlog() Facilities

Constant	Description
LOG_AUTH	security/authorization messages (use LOG_AUTHPRIV instead in systems where that constant is defined)
LOG_AUTHPRIV	security/authorization messages (private)
LOG_CRON	clock daemon (cron and at)
LOG_DAEMON	other system daemons
LOG_KERN	kernel messages
LOG_LOCAL0 ... LOG_LOCAL7	reserved for local use
LOG_LPR	line printer subsystem
LOG_MAIL	mail subsystem
LOG_NEWS	USENET news subsystem
LOG_SYSLOG	messages generated internally by syslogd

Constant	Description
LOG_USER	generic user-level messages
LOG_UUCP	UUCP subsystem

See also **define_syslog_variables()**, **syslog()** and **closelog()**.

pfsockopen (PHP 3>= 3.0.7, PHP 4)

Open persistent Internet or Unix domain socket connection

```
int pfsockopen (string hostname, int port [, int errno [, string errstr [, int timeout]]])
```

This function behaves exactly as **fsockopen()** with the difference that the connection is not closed after the script finishes. It is the persistent version of **fsockopen()**.

socket_get_status (PHP 4 >= 4.0b4)

Returns information about existing socket resource

```
array socket_get_status (resource socket_get_status)
```

Returns information about an existing socket resource. Currently returns four entries in the result array:

- *timed_out* (bool) - The socket timed out waiting for data
- *blocked* (bool) - The socket was blocked
- *eof* (bool) - Indicates EOF event
- *unread_bytes* (int) - Number of bytes left in the socket buffer

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, and **strerror()**.

socket_set_blocking (PHP 4 >= 4.0b4)

Set blocking/non-blocking mode on a socket

```
int socket_set_blocking (int socket descriptor, int mode)
```

If *mode* is false, the given socket descriptor will be switched to non-blocking mode, and if true, it will be switched to blocking mode. This affects calls like **fgets()** that read from the socket. In non-blocking mode an **fgets()** call will always return right away while in blocking mode it will wait for data to become available on the socket.

This function was previously called as **set_socket_blocking()** but this usage is deprecated.

socket_set_timeout (PHP 4 >= 4.0b4)

Set timeout period on a socket

```
bool socket_set_timeout (int socket descriptor, int seconds, int microseconds)
```

Sets the timeout value on *socket descriptor*, expressed in the sum of *seconds* and *microseconds*.

Příklad 1. socket_set_timeout() Example

```
<?php
$fp = fsockopen("www.php.net", 80);
if(!$fp) {
    echo "Unable to open\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

This function was previously called as **set_socket_timeout()** but this usage is deprecated.

See also: **fsockopen()** and **fopen()**.

syslog (PHP 3, PHP 4)

Generate a system log message

```
int syslog (int priority, string message)
```

Syslog() generates a log message that will be distributed by the system logger. *priority* is a combination of the facility and the level, values for which are given in the next section. The remaining argument is the message to send, except that the two characters %m will be replaced by the error message string (strerror) corresponding to the present value of errno.

Tabulka 1. Syslog() Priorities (in descending order)

Constant	Description
LOG_EMERG	system is unusable
LOG_ALERT	action must be taken immediately
LOG_CRIT	critical conditions
LOG_ERR	error conditions
LOG_WARNING	warning conditions
LOG_NOTICE	normal, but significant, condition
LOG_INFO	informational message
LOG_DEBUG	debug-level message

Příklad 1. Using syslog()

```
<?php
define_syslog_variables();
// open syslog, include the process ID and also send
// the log to standard error, and use a user defined
// logging mechanism
openlog("myScripLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);
```

```
// some code

if (authorized_client()) {
    // do something
} else {
    // unauthorized client!
    // log the attempt
    $access = date("Y/m/d H:i:s");
    syslog(LOG_WARNING,"Unauthorized client: $access $REMOTE_ADDR ($HTTP_USER_AGENT)");
}

closelog();
?>
```

For information on setting up a user defined log handler, see the syslog.conf(5) Unix manual page. More information on the syslog facilities and option can be found in the man pages for syslog(3) on Unix machines.

On Windows NT, the syslog service is emulated using the Event Log.

See also **define_syslog_variables()**, **openlog()** and **closelog()**.

L. Unified ODBC functions

In addition to normal ODBC support, the Unified ODBC functions in PHP allow you to access several databases that have borrowed the semantics of the ODBC API to implement their own API. Instead of maintaining multiple database drivers that were all nearly identical, these drivers have been unified into a single set of ODBC functions.

The following databases are supported by the Unified ODBC functions: Adabas D (http://www.adabas.com/), IBM DB2 (http://www.ibm.com/db2/), iODBC (http://www.iodbc.org/), Solid (http://www.solidtech.com/), and Sybase SQL Anywhere (http://www.sybase.com/).

> **Poznámka:** There is no ODBC involved when connecting to the above databases. The functions that you use to speak natively to them just happen to share the same names and syntax as the ODBC functions. The exception to this is iODBC. Building PHP with iODBC support enables you to use any ODBC-compliant drivers with your PHP applications. iODBC is maintained by OpenLink Software (http://www.openlinksw.com/). More information on iODBC, as well as a HOWTO, is available at www.iodbc.org (http://www.iodbc.org/).

odbc_autocommit (PHP 3>= 3.0.6, PHP 4)

Toggle autocommit behaviour

```
int odbc_autocommit (int connection_id [, int OnOff])
```

Without the *OnOff* parameter, this function returns auto-commit status for *connection_id*. True is returned if auto-commit is on, false if it is off or an error occurs.

If *OnOff* is true, auto-commit is enabled, if it is false auto-commit is disabled. Returns `true` on success, `false` on failure.

By default, auto-commit is on for a connection. Disabling auto-commit is equivalent with starting a transaction.

See also **odbc_commit()** and **odbc_rollback()**.

odbc_binmode (PHP 3>= 3.0.6, PHP 4)

Handling of binary column data

```
int odbc_binmode (int result_id, int mode)
```

(ODBC SQL types affected: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Passthru BINARY data
- ODBC_BINMODE_RETURN: Return as is
- ODBC_BINMODE_CONVERT: Convert to char and return

When binary SQL data is converted to character C data, each byte (8 bits) of source data is represented as two ASCII characters. These characters are the ASCII character representation of the number in its hexadecimal form. For example, a binary 00000001 is converted to `"01"` and a binary 11111111 is converted to `"FF"`.

Tabulka 1. LONGVARBINARY handling

binmode	longreadlen	result
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	return as is
ODBC_BINMODE_CONVERT	>0	return as char

If **odbc_fetch_into()** is used, passthru means that an empty string is returned for these columns.

If *result_id* is `0`, the settings apply as default for new results.

Poznámka: Default for longreadlen is `4096` and binmode defaults to `ODBC_BINMODE_RETURN`. Handling of binary long columns is also affected by **odbc_longreadlen()**

odbc_close (PHP 3>= 3.0.6, PHP 4)

Close an ODBC connection

```
void odbc_close (int connection_id)
```

odbc_close() will close down the connection to the database server associated with the given connection identifier.

Poznámka: This function will fail if there are open transactions on this connection. The connection will remain open in this case.

odbc_close_all (PHP 3>= 3.0.6, PHP 4)

Close all ODBC connections

```
void odbc_close_all(void);
```

odbc_close_all() will close down all connections to database server(s).

Poznámka: This function will fail if there are open transactions on a connection. This connection will remain open in this case.

odbc_commit (PHP 3>= 3.0.6, PHP 4)

Commit an ODBC transaction

```
int odbc_commit (int connection_id)
```

Returns: `true` on success, `false` on failure. All pending transactions on *`connection_id`* are committed.

odbc_connect (PHP 3>= 3.0.6, PHP 4)

Connect to a datasource

```
int odbc_connect (string dsn, string user, string password [, int cursor_type])
```

Returns an ODBC connection id or 0 (`false`) on error.

The connection id returned by this functions is needed by other ODBC functions. You can have multiple connections open at once. The optional fourth parameter sets the type of cursor to be used for this connection. This parameter is not normally needed, but can be useful for working around problems with some ODBC drivers.

With some ODBC drivers, executing a complex stored procedure may fail with an error similar to: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Using SQL_CUR_USE_ODBC may avoid that error. Also, some drivers don’t support the optional row_number parameter in **odbc_fetch_row()**. SQL_CUR_USE_ODBC might help in that case, too.

The following constants are defined for cursortype:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

For persistent connections see **odbc_pconnect()**.

odbc_cursor (PHP 3>= 3.0.6, PHP 4)

Get cursorname

```
string odbc_cursor (int result_id)
```

odbc_cursor will return a cursorname for the given result_id.

odbc_do (PHP 3>= 3.0.6, PHP 4)

Synonym for **odbc_exec()**

```
int odbc_do (int conn_id, string query)
```

Odbc_do() will execute a query on the given connection.

odbc_error (PHP 4 CVS only)

Get the last error code

```
string odbc_error ([int connection_id])
```

Returns a six-digit ODBC state, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: **odbc_errormsg()** and **odbc_exec()**.

odbc_errormsg (PHP 4 CVS only)

Get the last error message

```
string odbc_errormsg ([int connection_id])
```

Returns a string containing the last ODBC error message, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: **odbc_error()** and **odbc_exec()**.

odbc_exec (PHP 3>= 3.0.6, PHP 4)

Prepare and execute a SQL statement

```
int odbc_exec (int connection_id, string query_string)
```

Returns `false` on error. Returns an ODBC result identifier if the SQL command was executed successfully.

odbc_exec() will send an SQL statement to the database server specified by *`connection_id`*. This parameter must be a valid identifier returned by **odbc_connect()** or **odbc_pconnect()**.

See also: **odbc_prepare()** and **odbc_execute()** for multiple execution of SQL statements.

odbc_execute (PHP 3>= 3.0.6, PHP 4)

Execute a prepared statement

```
int odbc_execute (int result_id [, array parameters_array])
```

Executes a statement prepared with **odbc_prepare()**. Returns `true` on successful execution, `false` otherwise. The array *`arameters_array`* only needs to be given if you really have parameters in your statement.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4)

Fetch one result row into array

```
int odbc_fetch_into (int result_id [, int rownumber, array result_array])
```

Returns the number of columns in the result; `false` on error. *`result_array`* must be passed by reference, but it can be of any type since it will be converted to type array. The array will contain the column values starting at array index 0.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4)

Fetch a row

```
int odbc_fetch_row (int result_id [, int row_number])
```

If **odbc_fetch_row()** was succesful (there was a row), `true` is returned. If there are no more rows, `false` is returned.

odbc_fetch_row() fetches a row of the data that was returned by **odbc_do()** / **odbc_exec()**. After **odbc_fetch_row()** is called, the fields of that row can be accessed with **odbc_result()**.

If *`row_number`* is not specified, **odbc_fetch_row()** will try to fetch the next row in the result set. Calls to **odbc_fetch_row()** with and without *`row_number`* can be mixed.

To step through the result more than once, you can call **odbc_fetch_row()** with *`row_number`* 1, and then continue doing **odbc_fetch_row()** without *`row_number`* to review the result. If a driver doesn't support fetching rows by number, the *`row_number`* parameter is ignored.

odbc_field_name (PHP 3>= 3.0.6, PHP 4)

Get the columnname

```
string odbc_field_name (int result_id, int field_number)
```

odbc_field_name() will return the name of the field occupying the given column number in the given ODBC result identifier. Field numbering starts at 1. `false` is returned on error.

odbc_field_num (PHP 3>= 3.0.6, PHP 4)

Return column number

```
int odbc_field_num (int result_id, string field_name)
```

odbc_field_num() will return the number of the column slot that corresponds to the named field in the given ODBC result identifier. Field numbering starts at 1. `false` is returned on error.

odbc_field_type (PHP 3>= 3.0.6, PHP 4)

Datatype of a field

```
string odbc_field_type (int result_id, int field_number)
```

odbc_field_type() will return the SQL type of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

odbc_field_len (PHP 3>= 3.0.6, PHP 4)

Get the length (precision) of a field

```
int odbc_field_len (int result_id, int field_number)
```

odbc_field_len() will return the length of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

See also: **odbc_field_scale()** to get the scale of a floating point number.

odbc_field_precision (PHP 4 >= 4.0.0)

Synonym for **odbc_field_len()**

```
string odbc_field_precision (int result_id, int field_number)
```

odbc_field_precision() will return the precision of the field referecend by number in the given ODBC result identifier.

See also: **odbc_field_scale()** to get the scale of a floating point number.

odbc_field_scale (PHP 4 >= 4.0.0)

Get the scale of a field

```
string odbc_field_scale (int result_id, int field_number)
```

odbc_field_precision() will return the scale of the field referecend by number in the given ODBC result identifier.

odbc_free_result (PHP 3>= 3.0.6, PHP 4)

Free resources associated with a result

```
int odbc_free_result (int result_id)
```

Always returns `true`.

odbc_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **odbc_free_result()**, and the memory associated with *`result_id`* will be freed.

> **Poznámka:** If auto-commit is disabled (see **odbc_autocommit()**) and you call **odbc_free_result()** before commiting, all pending transactions are rolled back.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4)

Handling of LONG columns

```
int odbc_longreadlen (int result_id, int length)
```

(ODBC SQL types affected: LONG, LONGVARBINARY) The number of bytes returned to PHP is controled by the parameter length. If it is set to 0, Long column data is passed thru to the client.

> **Poznámka:** Handling of LONGVARBINARY columns is also affected by **odbc_binmode()**.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4)

Number of columns in a result

```
int odbc_num_fields (int result_id)
```

Odbc_num_fields() will return the number of fields (columns) in an ODBC result. This function will return -1 on error. The argument is a valid result identifier returned by **odbc_exec()**.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4)

Open a persistent database connection

```
int odbc_pconnect (string dsn, string user, string password [, int cursor_type])
```

Returns an ODBC connection id or 0 (`false`) on error. This function is much like **odbc_connect()**, except that the connection is not really closed when the script has finished. Future requests for a connection with the same *`dsn`*, *`user`*, *`password`* combination (via **odbc_connect()** and **odbc_pconnect()**) can reuse the persistent connection.

Poznámka: Persistent connections have no effect if PHP is used as a CGI program.

For information about the optional cursor_type parameter see the **odbc_connect()** function. For more information on persistent connections, refer to the PHP FAQ.

odbc_prepare (PHP 3>= 3.0.6, PHP 4)

Prepares a statement for execution

```
int odbc_prepare (int connection_id, string query_string)
```

Returns `false` on error.

Returns an ODBC result identifier if the SQL command was prepared successfully. The result identifier can be used later to execute the statement with **odbc_execute()**.

odbc_num_rows (PHP 3>= 3.0.6, PHP 4)

Number of rows in a result

```
int odbc_num_rows (int result_id)
```

odbc_num_rows() will return the number of rows in an ODBC result. This function will return -1 on error. For INSERT, UPDATE and DELETE statements **odbc_num_rows()** returns the number of rows affected. For a SELECT clause this `can` be the number of rows available.

Note: Using **odbc_num_rows()** to determine the number of rows available after a SELECT will return -1 with many drivers.

odbc_result (PHP 3>= 3.0.6, PHP 4)

Get result data

```
string odbc_result (int result_id, mixed field)
```

Returns the contents of the field.

`field` can either be an integer containing the column number of the field you want; or it can be a string containing the name of the field. For example:

```
$item_3 = odbc_result ($Query_ID, 3);
$item_val = odbc_result ($Query_ID, "val");
```

The first call to **odbc_result()** returns the value of the third field in the current record of the query result. The second function call to **odbc_result()** returns the value of the field whose field name is "val" in the current record of the query result. An error occurs if a column number parameter for a field is less than one or exceeds the number of columns (or fields) in the current record. Similarly, an error occurs if a field with a name that is not one of the fieldnames of the table(s) that is(are) being queried.

Field indices start from 1. Regarding the way binary or long column data is returned refer to **odbc_binmode ()** and **odbc_longreadlen()**.

odbc_result_all (PHP 3>= 3.0.6, PHP 4)

Print result as HTML table

```
int odbc_result_all (int result_id [, string format])
```

Returns the number of rows in the result or `false` on error.

Odbc_result_all() will print all rows from a result identifier produced by **odbc_exec()**. The result is printed in HTML table format. With the optional string argument *`format`*, additional overall table formatting can be done.

odbc_rollback (PHP 3>= 3.0.6, PHP 4)

Rollback a transaction

```
int odbc_rollback (int connection_id)
```

Rolls back all pending statements on *`connection_id`*. Returns `true` on success, `false` on failure.

odbc_setoption (PHP 3>= 3.0.6, PHP 4)

Adjust ODBC settings. Returns false if an error occurs, otherwise true.

```
int odbc_setoption (int id, int function, int option, int param)
```

This function allows fiddling with the ODBC options for a particular connection or query result. It was written to help find work arounds to problems in quirky ODBC drivers. You should probably only use this function if you are an ODBC programmer and understand the effects the various options will have. You will certainly need a good ODBC reference to explain all the different options and values that can be used. Different driver versions support different options.

Because the effects may vary depending on the ODBC driver, use of this function in scripts to be made publicly available is strongly discouraged. Also, some ODBC options are not available to this function because they must be set before the connection is established or the query is prepared. However, if on a particular job it can make PHP work so your boss doesn't tell you to use a commercial product, that's all that really matters.

`ID` is a connection id or result id on which to change the settings.For SQLSetConnectOption(), this is a connection id. For SQLSetStmtOption(), this is a result id.

`Function` is the ODBC function to use. The value should be 1 for SQLSetConnectOption() and 2 for SQLSetStmtOption().

Parameter *`option`* is the option to set.

Parameter *`param`* is the value for the given *`option`*.

Příklad 1. ODBC Setoption Examples

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    This example has the same effect as
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    This example sets the query to timeout after 30 seconds.

$result = odbc_prepare ($conn, $sql);
```

```
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_tables (PHP 3>= 3.0.17, PHP 4 >= 4.0b4)

Get the list of table names stored in a specific data source. Returns a result identifier containing the information.

```
int odbc_tables (int connection_id [, string qualifier [, string owner [, string name
[, string types]]]])
```

Lists all tables in the requested range. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

The result set is ordered by TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

To support enumeration of qualifiers, owners, and table types, the following special semantics for the *qualifier*, *owner*, *name*, and *table_type* are available:

- If *qualifier* is a single percent character (%) and *owner* and *name* are empty strings, then the result set contains a list of valid qualifiers for the data source. (All columns except the TABLE_QUALIFIER column contain NULLs.)
- If *owner* is a single percent character (%) and *qualifier* and *name* are empty strings, then the result set contains a list of valid owners for the data source. (All columns except the TABLE_OWNER column contain NULLs.)
- If *table_type* is a single percent character (%) and *qualifier*, *owner* and *name* are empty strings, then the result set contains a list of valid table types for the data source. (All columns except the TABLE_TYPE column contain NULLs.)

If *table_type* is not an empty string, it must contain a list of comma-separated values for the types of interest; each value may be enclosed in single quotes (') or unquoted. For example, "'TABLE','VIEW'" or "TABLE, VIEW". If the data source does not support a specified table type, **odbc_tables()** does not return any results for that type.

See also **odbc_tableprivileges()** to retrieve associated privileges.

odbc_tableprivileges (PHP 4 >= 4.0b4)

Lists tables and the privileges associated with each table

```
int odbc_tableprivileges (int connection_id [, string qualifier [, string owner [,
string name]]])
```

Lists tables in the requested range and the privileges associated with each table. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *`owner`* and *`name`* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_columns (PHP 4 >= 4.0b4)

Lists the column names in specified tables. Returns a result identifier containing the information.

```
int odbc_columns (int connection_id [, string qualifier [, string owner [, string
table_name [, string column_name]]]])
```

Lists all columns in the requested range. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *`owner`*, *`table_name`* and *`column_name`* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

See also **odbc_columnprivileges()** to retrieve associated privileges.

odbc_columnprivileges (PHP 4 >= 4.0b4)

Returns a result identifier that can be used to fetch a list of columns and associated privileges

```
int odbc_columnprivileges (int connection_id [, string qualifier [, string owner [,
string table_name [, string column_name]]]])
```

Lists columns and associated privileges for the given table. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *`column_name`* argument accepts search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_gettypeinfo (PHP 4 >= 4.0b4)

Returns a result identifier containing information about data types supported by the data source.

```
int odbc_gettypeinfo (int connection_id [, int data_type])
```

Retrieves information about data types supported by the data source. Returns an ODBC result identifier or `false` on failure. The optional argument *`data_type`* can be used to restrict the information to a single data type.

The result set has the following columns:

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE

- MAXIMUM_SCALE

The result set is ordered by DATA_TYPE and TYPE_NAME.

odbc_primarykeys (PHP 4 >= 4.0b4)

Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table

```
int odbc_primarykeys (int connection_id, string qualifier, string owner, string table)
```

Returns the column names that comprise the primary key for a table. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_foreignkeys (PHP 4 >= 4.0b4)

Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table

```
int odbc_foreignkeys (int connection_id, string pk_qualifier, string pk_owner, string
pk_table, string fk_qualifier, string fk_owner, string fk_table)
```

Odbc_foreignkeys() retrieves information about foreign keys. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

If *pk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing the primary key of the specified table and all of the foreign keys that refer to it.

If *fk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *pk_table* and *fk_table* contain table names, **odbc_foreignkeys()** returns the foreign keys in the table specified in *fk_table* that refer to the primary key of the table specified in *pk_table*. This should be one key at most.

odbc_procedures (PHP 4 >= 4.0b4)

Get the list of procedures stored in a specific data source. Returns a result identifier containing the information.

```
int odbc_procedures (int connection_id [, string qualifier [, string owner [, string
name]]])
```

Lists all procedures in the requested range. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_procedurecolumns (PHP 4 >= 4.0b4)

Retrieve information about parameters to procedures

```
int odbc_procedurecolumns (int connection_id [, string qualifier [, string owner [,
string proc [, string column]]]])
```

Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE

- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME and COLUMN_TYPE.

The *owner*, *proc* and *column* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_specialcolumns (PHP 4 >= 4.0b4)

Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction

```
int odbc_specialcolumns (int connection_id, int type, string qualifier, string owner,
string table, int scope, int nullable)
```

When the type argument is SQL_BEST_ROWID, **odbc_specialcolumns()** returns the column or columns that uniquely identify each row in the table.

When the type argument is SQL_ROWVER, **odbc_specialcolumns()** returns the optimal column or set of columns that, by retrieving values from the column or columns, allows any row in the specified table to be uniquely identified.

Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

The result set is ordered by SCOPE.

odbc_statistics (PHP 4 >= 4.0b4)

Retrieve statistics about a table

```
int odbc_statistics (int connection_id, string qualifier, string owner, string
table_name, int unique, int accuracy)
```

Get statistics about a table and it's indexes. Returns an ODBC result identifier or `false` on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE
- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

The result set is ordered by NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME and SEQ_IN_INDEX.

LI. Oracle 8 functions

These functions allow you to access Oracle8 and Oracle7 databases. It uses the Oracle8 Call-Interface (OCI8). You will need the Oracle8 client libraries to use this extension.

This extension is more flexible than the standard Oracle extension. It supports binding of global and local PHP variables to Oracle placeholders, has full LOB, FILE and ROWID support and allows you to use user-supplied define variables.

Before using this extension, make sure that you have set up your oracle environment variables properly for the Oracle user, as well as your web daemon user. The variables you might need to set are as follows:

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

After setting up the environment variables for your webserver user, be sure to also add the webserver user (nobody, www) to the oracle group.

If your webserver doesn't start or crashes at startup: Check that Apache is linked with the pthread library:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

If the libpthread is not listed you have to reinstall Apache:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Příklad 1. OCI Hints

```
<?php
// by sergo@bacup.ru

// Use option: OCI_DEFAULT for execute command to delay execution
OCIExecute($stmt, OCI_DEFAULT);

// for retrieve data use (after fetch):

$result = OCIResult($stmt, $n);
if (is_object ($result)) $result = $result->load();

// For INSERT or UPDATE statement use:
```

```
$sql = "insert into table (field1, field2) values (field1 = 'value',
 field2 = empty_clob()) returning field2 into :field2";
OCIParse($conn, $sql);
$clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName ($stmt, ":field2", &$clob, -1, OCI_B_CLOB);
OCIExecute($stmt, OCI_DEFAULT);
$clob->save ("some text");

?>
```

You can easily access stored procedures in the same way as you would from the commands line.

Příklad 2. Using Stored Procedures

```
<?php
// by webmaster@remoterealty.com
$sth = OCIParse ( $dbh, "begin sp_newaddress( :address_id, '$firstname',
 '$lastname', '$company', '$address1', '$address2', '$city', '$state',
 '$postalcode', '$country', :error_code );end;" );

// This calls stored procedure sp_newaddress, with :address_id being an
// in/out variable and :error_code being an out variable.
// Then you do the binding:

   OCIBindByName ( $sth, ":address_id", $addr_id, 10 );
   OCIBindByName ( $sth, ":error_code", $errorcode, 10 );
   OCIExecute ( $sth );

?>
```

OCIDefineByName (PHP 3>= 3.0.7, PHP 4)

Use a PHP variable for the define-step during a SELECT

```
int OCIDefineByName (int stmt, string Column-Name, mixed variable [, int type])
```

OCIDefineByName() uses fetches SQL-Columns into user-defined PHP-Variables. Be careful that Oracle user ALL-UPPERCASE column-names, whereby in your select you can also write lower-case. **OCIDefineByName()** expects the *Column-Name* to be in uppercase. If you define a variable that doesn't exists in you select statement, no error will be given!

If you need to define an abstract Datatype (LOB/ROWID/BFILE) you need to allocate it first using **OCINewDescriptor()** function. See also the **OCIBindByName()** function.

Příklad 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@thieso.net (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* the define MUST be done BEFORE ociexecute! */

OCIDefineByName($stmt,"EMPNO",$empno);
OCIDefineByName($stmt,"ENAME",$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIBindByName (PHP 3>= 3.0.4, PHP 4)

Bind a PHP variable to an Oracle Placeholder

```
int OCIBindByName (int stmt, string ph_name, mixed &variable, int length [, int type])
```

OCIBindByName() binds the PHP variable *variable* to the Oracle placeholder *ph_name*. Whether it will be used for input or output will be determined run-time, and the necessary storage space will be allocated. The *length* parameter sets the maximum length for the bind. If you set *length* to -1 **OCIBindByName()** will use the current length of *variable* to set the maximum length.

If you need to bind an abstract Datatype (LOB/ROWID/BFILE) you need to allocate it first using **OCINewDescriptor()** function. The *length* is not used for abstract Datatypes and should be set to -1. The *type* variable tells oracle, what kind of descriptor we want to use. Possible values are: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Příklad 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@thieso.net (980221)
```

```
  inserts 3 records into emp, and uses the ROWID for updating the
  records just after the insert.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
   "values (:empno,:ename) ".
   "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
OCIExecute($stmt);
OCIExecute($update);
}

$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

Varování

It is a bad idea to use magic quotes and **OciBindByName()** simultaneously as no quoting is needed on quoted variables and any quotes magically applied will be written into your database as **OciBindByName()** is not able to distinguish magically added quotings from those added by intention.

OCILogon (PHP 3>= 3.0.4, PHP 4)

Establishes a connection to Oracle

```
int OCILogon (string username, string password [, string db])
```

OCILogon() returns an connection identifier needed for most other OCI calls. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

Connections are shared at the page level when using **OCILogon()**. This means that commits and rollbacks apply to all open transactions in the page, even if you have created multiple connections.

This example demonstrates how the connections are shared.

Příklad 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."---selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."---done\n\n";
}

create_table($c1);
insert_data($c1);   // Insert a row using c1
insert_data($c2);   // Insert a row using c2
```

```
select_data($c1);   // Results of both inserts are returned
select_data($c2);

rollback($c1);      // Rollback using c1

select_data($c1);   // Both inserts have been rolled back
select_data($c2);

insert_data($c2);   // Insert a row using c2
commit($c2);        // commit using c2

select_data($c1);   // result of c2 insert is returned

delete_data($c1);   // delete all rows in table using c1
select_data($c1);   // no rows returned
select_data($c2);   // no rows returned
commit($c1);        // commit using c1

select_data($c1);   // no rows returned
select_data($c2);   // no rows returned


drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCIPLogon()** and **OCINLogon()**.

OCIPLogon (PHP 3>= 3.0.8, PHP 4)

Connect to an Oracle database and log on using a persistant connection. Returns a new session.

```
int OCIPLogon (string username, string password [, string db])
```

OCIPLogon() creates a persistent connection to an Oracle 8 database and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

See also **OCILogon()** and **OCINLogon()**.

OCINLogon (PHP 3>= 3.0.8, PHP 4)

Connect to an Oracle database and log on using a new connection. Returns a new session.

```
int OCINLogon (string username, string password [, string db])
```

OCINLogon() creates a new connection to an Oracle 8 database and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

OCINLogon() forces a new connection. This should be used if you need to isolate a set of transactions. By default, connections are shared at the page level if using **OCILogon()** or at the web server process level if using **OCIPLogon()**. If you have multiple connections open using **OCINLogon()**, all commits and rollbacks apply to the specified connection only.

This example demonstrates how the connections are separated.

Příklad 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."---selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."---done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
```

```
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);


drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCILogon()** and **OCIPLogon()**.

OCILogOff (PHP 3>= 3.0.4, PHP 4)

Disconnects from Oracle

```
int OCILogOff (int connection)
```

OCILogOff() closes an Oracle connection.

OCIExecute (PHP 3>= 3.0.4, PHP 4)

Execute a statement

```
int OCIExecute (int statement [, int mode])
```

OCIExecute() executes a previously parsed statement. (see **OCIParse()**. The optional *mode* allows you to specify the execution-mode (default is OCI_COMMIT_ON_SUCCESS). If you don't want statements to be committed automaticly specify OCI_DEFAULT as your mode.

OCICommit (PHP 3>= 3.0.7, PHP 4)

Commits outstanding transactions

```
int OCICommit (int connection)
```

OCICommit() commits all outstanding statements for Oracle connection *connection*.

OCIRollback (PHP 3>= 3.0.7, PHP 4)

Rolls back outstanding transactions

```
int OCIRollback (int connection)
```

OCIRollback() rolls back all outstanding statements for Oracle connection *connection*.

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4)

Initialize a new empty descriptor LOB/FILE (LOB is default)

```
string OCINewDescriptor (int connection [, int type])
```

OCINewDescriptor() Allocates storage to hold descriptors or LOB locators. Valid values for the valid *type* are OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. For LOB descriptors, the methods load, save, and savefile are associated with the descriptor, for BFILE only the load method exists. See the second example usage hints.

Příklad 1. OCINewDescriptor

```
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>

<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php3" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {
     // $lob_upload contains the temporary filename of the uploaded file
     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob)
               values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
```

```
     OCIExecute($stmt);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn't upload Blob\n";
     }
     OCIFreeDesc($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
?>
```

OCIRowCount (PHP 3>= 3.0.7, PHP 4)

Gets the number of affected rows

```
int OCIRowCount (int statement)
```

OCIRowCount() returns the number of rows affected for eg update-statements. This function will not tell you the number of rows that a select will return!

Příklad 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCINumCols (PHP 3>= 3.0.4, PHP 4)

Return the number of result columns in a statement

```
int OCINumCols (int stmt)
```

OCINumCols() returns the number of columns in a statement

Příklad 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
```

```
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIResult (PHP 3>= 3.0.4, PHP 4)

Returns column value for fetched row

```
mixed OCIResult (int statement, mixed column)
```

OCIResult() returns the data for column *column* in the current row (see **OCIFetch()**).**OCIResult()** will return everything as strings except for abstract types (ROWIDs, LOBs and FILEs).

OCIFetch (PHP 3>= 3.0.4, PHP 4)

Fetches the next row into result-buffer

```
int OCIFetch (int statement)
```

OCIFetch() fetches the next row (for SELECT statements) into the internal result-buffer.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4)

Fetches the next row into result-array

```
int OCIFetchInto (int stmt, array &result [, int mode])
```

OCIFetchInto() fetches the next row (for SELECT statements) into the *result* array. **OCIFetchInto()** will overwrite the previous content of *result*. By default *result* will contain a one-based array of all columns that are not NULL.

The *mode* parameter allows you to change the default behaviour. You can specify more than one flag by simply adding them up (eg OCI_ASSOC+OCI_RETURN_NULLS). The known flags are:

```
OCI_ASSOC Return an associative array.
OCI_NUM Return an numbered array starting with one. (DEFAULT)
OCI_RETURN_NULLS Return empty columns.
OCI_RETURN_LOBS Return the value of a LOB instead of the descriptor.
```

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4)

Fetch all rows of result data into an array.

```
int OCIFetchStatement (int stmt, array &variable)
```

OCIFetchStatement() fetches all the rows from a result into a user-defined array. **OCIFetchStatement()** returns the number of rows fetched.

Příklad 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4)

test whether a result column is NULL

```
int OCIColumnIsNULL (int stmt, mixed column)
```

OCIColumnIsNULL() returns true if the returned column *column* in the result from the statement *stmt* is NULL. You can either use the column-number (1-Based) or the column-name for the *col* parameter.

OCIColumnName (PHP 3>= 3.0.4, PHP 4)

Returns the name of a column.

```
string OCIColumnName (int stmt, int col)
```

OCIColumnName() returns the name of the column corresponding to the column number (1-based) that is passed in.

Příklad 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnType()**, and **OCIColumnSize()**.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4)

return result column size

```
int OCIColumnSize (int stmt, mixed column)
```

OCIColumnSize() returns the size of the column as given by Oracle. You can either use the column-number (1-Based) or the column-name for the *col* parameter.

Příklad 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
```

```
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnName()**, and **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4)

Returns the data type of a column.

```
mixed OCIColumnType (int stmt, int col)
```

OCIColumnType() returns the data type of the column corresponding to the column number (1-based) that is passed in.

Příklad 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
```

```
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnName()**, and **OCIColumnSize()**.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4)

Return a string containing server version information.

```
string OCIServerVersion (int conn)
```

Příklad 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Server Version: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

OCIStatementType (PHP 3>= 3.0.5, PHP 4)

Return the type of an OCI statement.

```
string OCIStatementType (int stmt)
```

OCIStatementType() returns one of the following values:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Příklad 1. Code examples

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
```

```
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCINewCursor (PHP 3>= 3.0.8, PHP 4)

Return a new cursor (Statement-Handle) - use to bind ref-cursors.

```
int OCINewCursor (int conn)
```

OCINewCursor() allocates a new statement handle on the specified connection.

Příklad 1. Using a REF CURSOR from a stored procedure

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Příklad 2. Using a REF CURSOR in a select statement

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
```

```
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data[ "EMPCNT" ]);
    while (OCIFetchInto($data[ "EMPCNT" ],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4)

Free all resources associated with a statement.

```
int OCIFreeStatement (int stmt)
```

OCIFreeStatement() returns true if successful, or false if unsuccessful.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4)

Free all resources associated with a cursor.

```
int OCIFreeCursor (int stmt)
```

OCIFreeCursor() returns true if successful, or false if unsuccessful.

OCIFreeDesc (PHP 4 >= 4.0b4)

Deletes a large object descriptor.

```
int OCIFreeDesc (object lob)
```

OCIFreeDesc() returns true if successful, or false if unsuccessful.

OCIParse (PHP 3>= 3.0.4, PHP 4)

Parse a query and return a statement

```
int OCIParse (int conn, strint query)
```

OCIParse() parses the *query* using *conn*. It returns the statement identity if the query is valid, false if not. The *query* can be any valid SQL statement.

OCIError (PHP 3>= 3.0.7, PHP 4)

Return the last error of stmt|conn|global. If no error happened returns false.

```
array OCIError ([int stmt|conn|global])
```

OCIError() returns the last error found. If the optional *stmt|conn|global* is not provided, the last error encountered is returned. If no error is found, **OCIError()** returns false. **OCIError()** returns the error as an associative array. In this array, *code* consists the oracle error code and *message* the oracle errorstring.

OCIInternalDebug (PHP 3>= 3.0.4, PHP 4)

Enables or disables internal debug output. By default it is disabled

```
void OCIInternalDebug (int onoff)
```

OCIInternalDebug() enables internal debug output. Set *onoff* to 0 to turn debug output off, 1 to turn it on.

LII. OpenSSL funkce

Tato extenze využívá funkce OpenSSL (http://www.openssl.org/) pro tvorbu a ověřování podpisů a pečetění (kódování) a otvírání (dekódování) dat. K práci s touto extenzí potřebujete OpenSSL >= 0.9.6.

OpenSSL nabízí mnoho vlastností, které tato extenze v současnosti nepodporuje. Některé z nich mohou být v budoucnu přidány.

openssl_free_key (PHP 4 >= 4.0.4)

Uvolnit prostředky klíče

```
void openssl_free_key (int key_identifier)
```

openssl_free_key() uvolní klíč asociovaný s předaným *key_identifier* z paměti.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Připravit soukromý PEM klíč k použití

```
int openssl_get_privatekey (string key [, string passphrase])
```

Pri úspěchu vrací klíč, při chybě `false`.

openssl_get_privatekey() rozparsuje sourkomý PEM klíč *key* a připraví ho k použití v dalších funkcích. Volitelný argument *passphrase* musí být předán, pokud je tento klíč zakódován (chráněn heslem).

openssl_get_publickey (PHP 4 >= 4.0.4)

Získat z certifikátu veřejný klíč a připravit ho k použití

```
int openssl_get_publickey (string certificate)
```

Při úspěchu vrací identifikátor klíče, při chybě `false`.

openssl_get_publickey() z X.509 certifikátu *certificate* vyextrahuje veřejný klíč a připraví ho k použití v dalších funkcích.

openssl_open (PHP 4 >= 4.0.4)

Otevřít zapečetěná data

```
bool openssl_open (string sealed_data, string open_data, string env_key, int
priv_key_id)
```

Při úspěchu vrací `true`, při chybě `false`. Úspěšně otevřená data se umístí do argumentu *open_data*.

openssl_open() otevře (dekóduje) *sealed_data* pomocí soukromého klíče asociovaného s identifikátorem *priv_key_id* a obálkou *env_key*. Tato obálka se generuje při pečetění dat a je použitelná pouze s jedním utčitým soukromým klíčem. Více informací viz **openssl_seal()**.

Příklad 1. Ukázka openssl_open()

```
// $sealed a $env_key obsahují zapečetěná data a obálku
// obojí nám bylo dáno tím, kdo data zapečetil

// získat ze souboru soukromý klíč a připravit ho
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// dekódovat data a uložit je v $open
```

```
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "tady jsou otevřená data: ", $open;
else
    echo "nepodařilo se otevřít data";

// uvolnit klíč z paměti
openssl_free_key($pkeyid);
```

Viz také **openssl_seal()**.

openssl_seal (PHP 4 >= 4.0.4)

Zapečetit (zakódovat) data

```
int openssl_seal (string data, string sealed_data, array env_keys, array pub_key_ids)
```

Při úspěchu vrací délku zapečetěných dat, při chybě `false`. Úspěšně zapečetěná data se umístí do argumentu *`sealed_data`*, a obálka do *`env_keys`*.

openssl_seal() zapečetí (zakóduje) *`data`* pomocí RC4 s náhodně generovaným tajným klíčem. Tento klíč se zakóduje všemi veřejnými klíči asociovanými s identifikátory v *`pub_key_ids`* a zakódované klíče se vrátí v *`env_keys`*. To znamená, že lze poslat zapečetěná data více příjemcům (za předpokladu, že máme jejích veřejné klíče). Každý z příjemců musí obdržet zapečetěná data a obálku, která byla zakódována jeho veřejným klíčem.

Příklad 1. Ukázka openssl_seal()

```
// $data obsahuje data určená k zapečetění

// získat veřejné klíče našich příjemců a připravit je
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// opakovat prodruhého příjemce
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// zapečetit zprávu, pouze majitelé $pk1 a $pk2 mohou dekódovat $sealed pomocí
// $ekeys[0] a $ekeys[1].
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// uvolnit klíče z paměti
openssl_free_key($pk1);
openssl_free_key($pk2);
```

Viz také **openssl_open()**.

openssl_sign (PHP 4 >= 4.0.4)

Generate signature

```
bool openssl_sign (string data, string signature, int priv_key_id)
```

Při úspěchu vrací `true`, při chybě `false`. Úspěšně vytvořený podpis se umístí v *`signature`*.

openssl_sign() vypočítá podpis pro daná specified *`data`* pomocí SHA1 hashe a následně jej zakóduje soukromým klíčem asociovaným s *`priv_key_id`*. Pozn.: samotná data nejsou kódována.

Příklad 1. openssl_sign() example

```
// $data obsahuje data určená k podpisu

// získat ze souboru soukromý klíč a připravit ho
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// vytvořit podpis
openssl_sign($data, $signature, $pkeyid);

// uvolnit klíč z paměti
openssl_free_key($pkeyid);
```

Viz také **openssl_verify()**.

openssl_verify (PHP 4 >= 4.0.4)

Ověřit podpis

```
int openssl_verify (string data, string signature, int pub_key_id)
```

Vrací 1, pokud je podpis správný, 0, pokud je nesprávný, a -1 při chybě.

openssl_verify() ověřuje, zda je *`signature`* správný pro *`data`* pomocí veřejného klíče asociovaného s *`pub_key_id`*. Musí to být veřejný klíč odpovídající soukromému klíči použitému k podpisu.

Příklad 1. Ukázka openssl_verify()

```
// $data a $signature obsahují data a podpis

// získat z certifikátu veřejný klíč a připravit ho
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// zjistit, jestli je podpis v pořádku
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "dobře";
elseif ($ok == 0)
    echo "špatně";
else
    echo "nejhůř, při kontrole podpisu došlo k chybě";

// uvolnit klíč z paměti
openssl_free_key($pubkeyid);
```

Viz také **openssl_sign()**.

LIII. Oracle functions

Ora_Bind (PHP 3, PHP 4)

bind a PHP variable to an Oracle parameter

```
int ora_bind (int cursor, string PHP variable name, string SQL parameter name, int
length [, int type])
```

Returns true if the bind succeeds, otherwise false. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function binds the named PHP variable with a SQL parameter. The SQL parameter must be in the form ":name". With the optional type parameter, you can define whether the SQL parameter is an in/out (0, default), in (1) or out (2) parameter. As of PHP 3.0.1, you can use the constants ORA_BIND_INOUT, ORA_BIND_IN and ORA_BIND_OUT instead of the numbers.

ora_bind must be called after **ora_parse()** and before **ora_exec()**. Input values can be given by assignment to the bound PHP variables, after calling **ora_exec()** the bound PHP variables contain the output values if available.

```
<?php
ora_parse($curs, "declare tmp INTEGER; be-
gin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close (PHP 3, PHP 4)

close an Oracle cursor

```
int ora_close (int cursor)
```

Returns true if the close succeeds, otherwise false. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function closes a data cursor opened with **ora_open()**.

Ora_ColumnName (PHP 3, PHP 4)

get name of Oracle result column

```
string Ora_ColumnName (int cursor, int column)
```

Returns the name of the field/column *column* on the cursor *cursor*. The returned name is in all uppercase letters.

Ora_ColumnSize (PHP 3, PHP 4)

get size of Oracle result column

```
int Ora_ColumnSize (int cursor, int column)
```

Returns the size of the Oracle column *column* on the cursor *cursor*.

Ora_ColumnType (PHP 3, PHP 4)

get type of Oracle result column

```
string Ora_ColumnType (int cursor, int column)
```

Returns the Oracle data type name of the field/column *column* on the cursor *cursor*. The returned type will be one of the following:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4)

commit an Oracle transaction

```
int ora_commit (int conn)
```

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function commits an Oracle transaction. A transaction is defined as all the changes on a given connection since the last commit/rollback, autocommit was turned off or when the connection was established.

Ora_CommitOff (PHP 3, PHP 4)

disable automatic commit

```
int ora_commitoff (int conn)
```

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function turns off automatic commit after each **ora_exec()**.

Ora_CommitOn (PHP 3, PHP 4)

enable automatic commit

```
int ora_commiton (int conn)
```

This function turns on automatic commit after each **ora_exec()** on the given connection.

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Do (PHP 3, PHP 4)

Parse, Exec, Fetch

```
int ora_do (int conn, string query)
```

This function is quick combination of **ora_parse()**, **ora_exec()** and **ora_fetch()**. It will parse and execute a statement, then fetch the first result row.

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

See also **ora_parse()**,**ora_exec()**, and **ora_fetch()**.

Ora_Error (PHP 3, PHP 4)

get Oracle error message

```
string Ora_Error (int cursor_or_connection)
```

Returns an error message of the form *XXX-NNNNN* where *XXX* is where the error comes from and *NNNNN* identifies the error message.

Poznámka: Support for connection ids was added in 3.0.4.

On UNIX versions of Oracle, you can find details about an error message like this: `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key`

Ora_ErrorCode (PHP 3, PHP 4)

get Oracle error code

```
int Ora_ErrorCode (int cursor_or_connection)
```

Returns the numeric error code of the last executed statement on the specified cursor or connection.

* *FIXME: should possible values be listed?*

Poznámka: Support for connection ids was added in 3.0.4.

Ora_Exec (PHP 3, PHP 4)

execute parsed statement on an Oracle cursor

```
int ora_exec (int cursor)
```

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

See also **ora_parse()**, **ora_fetch()**, and **ora_do()**.

Ora_Fetch (PHP 3, PHP 4)

fetch a row of data from a cursor

```
int ora_fetch (int cursor)
```

Returns true (a row was fetched) or false (no more rows, or an error occured). If an error occured, details can be retrieved using the **ora_error()** and **ora_errorcode()** functions. If there was no error, **ora_errorcode()** will return 0.

Retrieves a row of data from the specified cursor.

See also **ora_parse()**,**ora_exec()**, and **ora_do()**.

Ora_Fetch_Into (PHP 3, PHP 4)

Fetch a row into the specified result array

```
int ora_fetch_into (int cursor, array result [, int flags])
```

You can fetch a row into an array with this function.

Příklad 1. Oracle fetch into array

```
<?php
array($results);
ora_fetch_into($cursor, &$results);
echo $results[0];
echo $results[1];
?>
```

Note that you need to fetch the array by reference.

See also **ora_parse()**,**ora_exec()**, **ora_fetch()**, and **ora_do()**.

Ora_GetColumn (PHP 3, PHP 4)

get data from a fetched column

```
mixed ora_getcolumn (int cursor, mixed column)
```

Returns the column data. If an error occurs, False is returned and **ora_errorcode()** will return a non-zero value. Note, however, that a test for False on the results from this function may be true in cases where there is not error as well (NULL result, empty string, the number 0, the string "0").

Fetches the data for a column or function result.

Ora_Logoff (PHP 3, PHP 4)

close an Oracle connection

```
int ora_logoff (int connection)
```

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Logs out the user and disconnects from the server.

See also **ora_logon()**.

Ora_Logon (PHP 3, PHP 4)

open an Oracle connection

```
int ora_logon (string user, string password)
```

Establishes a connection between PHP and an Oracle database with the given username and password.

Connections can be made using SQL*Net by supplying the TNS name to *user* like this:

```
$conn = Ora_Logon("user@TNSNAME", "pass");
```

If you have character data with non-ASCII characters, you should make sure that NLS_LANG is set in your environment. For server modules, you should set it in the server's environment before starting the server.

Returns a connection index on success, or false on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_pLogon (PHP 3, PHP 4)

Open a persistent Oracle connection

```
int ora_plogon (string user, string password)
```

Establishes a persistant connection between PHP and an Oracle database with the given username and password.

See also **ora_logon()**.

Ora_Numcols (PHP 3, PHP 4)

Returns the number of columns

```
int ora_numcols (int cursor_ind)
```

ora_numcols() returns the number of columns in a result. Only returns meaningful values after an parse/exec/fetch sequence.

See also **ora_parse()**,**ora_exec()**, **ora_fetch()**, and **ora_do()**.

Ora_Numrows (PHP 3, PHP 4)

Returns the number of rows

```
int ora_numrows (int cursor_ind)
```

ora_numrows() returns the number of rows in a result.

Ora_Open (PHP 3, PHP 4)

open an Oracle cursor

```
int ora_open (int connection)
```

Opens an Oracle cursor associated with connection.

Returns a cursor index or False on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Parse (PHP 3, PHP 4)

parse an SQL statement

```
int ora_parse (int cursor_ind, string sql_statement, int defer)
```

This function parses an SQL statement or a PL/SQL block and associates it with the given cursor.

Returns 0 on success or -1 on error.

See also **ora_exec()**, **ora_fetch()**, and **ora_do()**.

Ora_Rollback (PHP 3, PHP 4)

roll back transaction

```
int ora_rollback (int connection)
```

This function undoes an Oracle transaction. (See **ora_commit()** for the definition of a transaction.)

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

LIV. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (http://www.ovrimos.com/). To enable ovrimos support in PHP just compile php with the '–with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Příklad 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

```
int ovrimos_connect (string host, string db, string user, string password)
```

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Příklad 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

```
void ovrimos_close (int connection)
```

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_close_all (PHP 4 >= 4.0.3)

Closes all the connections to ovrimos

```
void ovrimos_close_all (void)
```

ovrimos_close_all() is used to close all the connections.

ovrimos_close_all() closes all connections to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

```
int ovrimos_longreadlen (int result_id, int length)
```

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. Regardless of its taking a result_id as an argument, it currently sets this parameter for all result sets. Returns true.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

```
int ovrimos_prepare (int connection_id, string query)
```

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or false on failure).

Příklad 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

```
int ovrimos_execute (int result_id [, array parameters_array])
```

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns true or false. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don’t follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn’t bring myself on liking it.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

```
int ovrimos_cursor (int result_id)
```

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

```
int ovrimos_exec (int connection_id, string query)
```

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or false. Evidently, the SQL statement should not contain parameters.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

```
int ovrimos_fetch_into (int result_id, array result_array [, string how [, int
rownumber]])
```

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns true or false.

Příklad 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
```

```
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

```
int ovrimos_fetch_row (int result_id [, int how [, int row_number]])
```

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns true or false.

Příklad 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

```
int ovrimos_result (int result_id, mixed field)
```

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by ’field’, either as a string or as an 1-based index.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

```
int ovrimos_result_all (int result_id [, string format])
```

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns true or false.

Příklad 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Příklad 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i<=$colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

Příklad 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

```
int ovrimos_num_rows (int result_id)
```

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

```
int ovrimos_num_fields (int result_id)
```

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

```
int ovrimos_field_name (int result_id, int field_number)
```

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

```
int ovrimos_field_type (int result_id, int field_number)
```

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

```
int ovrimos_field_len (int result_id, int field_number)
```

ovrimos_field_len() is used to get the length of the output column.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

```
int ovrimos_field_num (int result_id, string field_name)
```

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or false.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

```
int ovrimos_free_result (int result_id)
```

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id. Returns true.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

```
int ovrimos_commit (int connection_id)
```

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

```
int ovrimos_rollback (int connection_id)
```

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LV. Output Control funkce

Output Control funkce (funkce pro řízení výstupu) vám umožňujíc ovládat, kdy se odešle výstup skriptu. To může být užitečné v několika různých situacích, zvláště pokud potřebujete poslat browseru hlavičky poté, co váš skript začal odesílat data. Output Control funkce neovlivňují hlavičky odeslané pomocí **header()** nebo **setcookie()**, pouze funkce jako **echo()** a data mezi bloky PHP kódu.

Příklad 1. Ukázka řízení výstupu

```
<?php

ob_start();
echo "Hello\n";

setcookie ("cookiename", "cookiedata");

ob_end_flush();

?>
```

Ve výše uvedené ukázce se výstup z **echo()** uloží ve výstupním bufferu až do volání **ob_end_flush()**. Mezitím volání **setcookie()** úspěšně uložilo cookie bez vyvolání chyby. (Normálně nemůžete odeslat do browseru hlavičky poté, co už byla odeslána data.)

Viz také **header()** a **setcookie()**.

flush (PHP 3, PHP 4)

Odeslat výstupní buffer

```
void flush(void);
```

Vyprázdní výstupní buffery PHP a jakéhokoli backendu, který PHP používá (CGI, web server, atd.) Odešle veškerý dosavadní výstup do uživatelova browseru.

> **Poznámka: flush()** nemá žádný účinek na bufferovací schéma vašeho webserveru nebo browseru na klientské straně.
>
> Některé servery, zvláště na Win32, bufferují výstup až do ukončení běhu skriptu bez ohledu na **flush()**, a až potom odešlou výstup browseru.
>
> Browser může také bufferovat svůj vstup před zobrazením. Například Netscape bufferuje text do té doby než přijme konec řádku nebo začátek tagu, a nezobrazí tabulku, dokud nedostane </table> nejzevnější tabulky.

ob_start (PHP 4)

Zapnout bufferování výstupu

```
void ob_start ([string output_callback])
```

Tato funkce zapíná bufferování výstupu. Pokud je bufferování výstupu aktivováno, žádný výstup ze skriptu se neodešle, místo toho se ukládá v interním bufferu.

Obsah tohoto interního bufferu je možno zkopírovat do proměnné typu string pomocí **ob_get_contents()**. K odeslání obsahu interního bufferu použijte **ob_end_flush()**. Naprotitomu **ob_end_clean()** tiše odstraní obsah výstupního bufferu.

Můžete zadat volitelný název callback funkce, která se automaticky zavolá s obsahem bufferu jako argumentem. Tato funkce musí přijímat řetězec a vracet řetězec. Tato funkce bude volána při **ob_end_flush()** a dostane obsah výstupního bufferu jako svůj argument. Musí vrátit nový výstupní buffer, který se pak vytiskne.

Výstupní buffery se dají stackovat, tzn. můžete zavolat **ob_start()** zatímco je aktivní další **ob_start()**. Je potřeba pouze správný počet volání **ob_end_flush()()**. Pokud je akivních více output callback funkcí, výstup je filtrován postupně přes každou z nich tak jak jsou do sebe vnořené.

Příklad 1. Ukázka callback funkce

```
<?php
function c($str) {
  // Druu Chunusun mut dum Kuntrubuß...
  return nl2br(ereg_replace("[aeiou]", "u", $str));
}

function d($str) {
  return strip_tags($str);
}
?>

<?php ob_start("c"); ?>
Drei Chinesen mit dem Kontrabaß...
<?php ob_start("d"); ?>
<h1>..saßen auf der Straße und erzählten sich was...</h1>
<?php ob_end_flush(); ?>
... da kam die Polizei, ja was ist denn das?
<?php ob_end_flush(); ?>

?>
```

Viz také **ob_get_contents()**, **ob_end_flush()**, **ob_end_clean()**, and **ob_implicit_flush()**

ob_get_contents (PHP 4)

Vrátit obsah výstupního bufferu

```
string ob_get_contents(void);
```

Tato funkce vrátí obsah výstupního bufferu nebo `false`, pokud bufferování výstupu není aktivováno.

Viz také **ob_start()** a **ob_get_length()**.

ob_get_length (PHP 4 >= 4.0.2)

Vrátit délku výstupního buffer

```
string ob_get_length(void);
```

Tato funkce vrátí délku obsahu výstupního bufferu nebo This will return the length of the contents in the output buffer, nebo `false`, pokud bufferování výstupu není aktivováno.

Viz také **ob_start()** a **ob_get_contents()**.

ob_end_flush (PHP 4)

Vyprázdnit (odeslat) výstupní buffer a vypnout bufferování výstupu

```
void ob_end_flush(void);
```

Tato funkce odešle obsah výstupního bufferu (pokud nějaký je) a vypne bufferování výstupu. Pokud chcete obsah výstupního bufferu dále zpracovávat, musíte před **ob_end_flush()** zavolat **ob_get_contents()**, protože obsah bufferu se po **ob_end_flush()** odhodí.

Viz také **ob_start()**, **ob_get_contents()** a **ob_end_clean()**.

ob_end_clean (PHP 4)

Vyčistit (vymazat) výstupní buffer a vypnout bufferování výstupu

```
void ob_end_clean(void);
```

Tato funkce odhodí obsah výstupního bufferu a vypne bufferování výstupu.

Viz také **ob_start()** a **ob_end_flush()**.

ob_implicit_flush (PHP 4 >= 4.0b4)

Vypnout/zapnout implicitní flush

```
void ob_implicit_flush ([int flag])
```

ob_implicit_flush() vypne nebo zapne implicitní flushování (pokud nedostane žádný `flag`, default je zapnout). Implicitní flushování způsobí flush po každém vygenerování výstupu, takže už nebudou potřeba explicitní volání **flush()**.

Zapnutí implicitního flushování zruší bufferování výstupu, a aktuální obsah výstupních bufferů se odešle jako při volání **ob_end_flush()**.

Viz také **flush()**, **ob_start()** a **ob_end_flush()**.

LVI. PDF funkce

Introduction

Pokud máte PDF knihovnu od Thomase Merze (dostupná z http://www.pdflib.com/pdflib/index.html), můžete používat PDF funkce na tvorbu PDF souborů; ke kompilaci budete potřebovat také JPEG knihovnu (ftp://ftp.uu.net/graphics/jpeg/) a TIFF knihovnu (http://www.libtiff.org/). Tyto dvě knihovny poměrně často dělají potíže při konfiguraci PHP. Při řešení případných problémů se řiďte chybovými zprávami configure skriptu.

Věnujte prosím pozornost výborné dokumentaci pdflib, která je součástí distribuce zdrojového kódu. Poskytuje velmi dobrý přehled schopností pdflib. Většina funkcí pdflib a příslušného PHP modulu má stejné jméno. Argumenty jsou také identické. Pokud chcete tento modul využívat opravdu efektivně, měli byste chápat také některé z konceptů PDF nebo Postscriptu. Všechny rozměry a koordináty se udávají v Postscriptových bodech. Obecně je 72 PostScriptových bodů na palec, ale závisí to na výstupním rozlišení.

Existuje další PHP modul na tvorbu PDF dokumentů, založený na ClibPDF od firmy FastIO (http://www.fastio.com/). Má mírně jinou API. Detaily viz ClibPDF funkce.

Tento PDF modul zavádí nový typ proměnné. Nazývá se *`pdfdoc`*. *`pdfdoc`* je pointer na PDF dokument a téměř všechny funkce ho vyžadují jako svůj první argument.

Zmatek se starými verzemi pdflib

Od úplného začátky podpory PDF v PHP — od pdflib 0.6 — došlo k mnoha změnám zvláště v API pdflib. Většinu těchto změn PHP nějak zakrylo, některé vyžadovaly změnu PHP API. Od pdflib 3.x se API snad stabilizovala, a PHP 4 přijala tuto verzi jako mimimální pro podporu PDF. Následkem toho mnoho funkcí dříve či později zmizí nebo bude nahraženo alternativami. Podpora pdflib 0.6 už byla naprosto ukončena. Následující tabulka vyjmenovává všechny funkce, které jsou od PHP 4.0.2 zastaralé a měly by být nahraženy jejich novějšími verzemi.

Tabulka 1. Zastaralé funkce a jejich náhrady

Stará funkce	Náhrada
pdf_put_image()	Není potřeba.
pdf_get_font()	**pdf_get_value()** s **`"font"`** jako druhý argument.
pdf_get_fontsize()	**pdf_get_value()** s **`"fontsize"`** jako druhý argument.
pdf_get_fontname()	**pdf_get_parameter()** s **`"fontname"`** jako druhý argument.
pdf_set_info_creator()	**pdf_set_info()** s **`"Creator"`** jako druhý argument.
pdf_set_info_title()	**pdf_set_info()** s **`"Title"`** jako druhý argument.
pdf_set_info_subject()	**pdf_set_info()** s **`"Subject"`** jako druhý argument.
pdf_set_info_author()	**pdf_set_info()** s **`"Author"`** jako druhý argument.
pdf_set_info_keywords()	**pdf_set_info()** s **`"Keywords"`** jako druhý argument.
pdf_set_leading()	**pdf_set_value()** s **`"leading"`** jako druhý argument.
pdf_set_text_rendering()	**pdf_set_value()** s **`"textrendering"`** jako druhý argument.
pdf_set_text_rise()	**pdf_set_value()** s **`"textrise"`** jako druhý argument.
pdf_set_horiz_scaling()	**pdf_set_value()** s **`"horizscaling"`** jako druhý argument.
pdf_set_text_matrix()	neexistuje

Stará funkce	Náhrada
pdf_set_char_spacing()	**pdf_set_value()** s **"charspacing" jako druhý argument.**
pdf_set_word_spacing()	**pdf_set_value()** s **"wordspacing" jako druhý argument.**
pdf_set_transition()	**pdf_set_parameter()** s **"transition" jako druhý argument.**
pdf_set_duration()	**pdf_set_value()** s **"duration" jako druhý argument.**
pdf_open_gif()	**pdf_open_image_file()** s **"gif" jako druhý argument.**
pdf_open_jpeg()	**pdf_open_image_file()** s **"jpeg" jako druhý argument.**
pdf_open_tiff()	**pdf_open_image_file()** s **"tiff" jako druhý argument.**
pdf_open_png()	**pdf_open_image_file()** s **"png" jako druhý argument.**
pdf_get_imagewidth()	**pdf_get_value()** s **"imagewidth" jako druhý argument a obrázkem jako třetí argument.**
pdf_get_imageheight()	**pdf_get_value()** s **"imageheight" jako druhý argument a obrázkem jako třetí argument.**
()	**()**

Rady pro instalaci pdflib 3.x

Od pdflib 3.0 by se pdflib měla konfigurovat s volbou `-enable-shared-pdflib`.

Problémy se staršími verzemi pdflib

Pokud používáte pdflib 2.01, zkontrolujte, jak je tato knihovna nainstalována. Měli byste mít soubor libpdf.so, nebo link na něj. Verze 2.01 vytváří soubor libpdf2.01.so, který se nedá najít při linkování testovacího souboru v configure. Budete muset vytvořit symbolický link z libpdf.so na libpdf2.01.so.

Ve verzi 2.20 přibyly další změny v API pdflib a podpora čínských a japonských fontů. Pokud používáte pdflib 2.20 buď'te opatrní při generování PDF dokumentů v paměti. Do verze pdflib 3.0 by mohlo být nestabilní. Argument kódování v **pdf_set_font()** se změnil na řetězec. To znamená, že místo např. 4 musíte použít 'winansi'.

Pokud používáte pdflib 2.30, nemáte k dispozici **pdf_set_text_matrix()**. Přestala být podporována. Obecnou radou je zjistit si případné změny v release notes používané verze pdflib.

Žádná verze PHP 4 od data 9. března 2000 nepodporuje podflib starší než 3.0. Na druhou stranu, PHP 3 by se nemělo používat s novější verzí pdflib než 2.01.

Ukázky

Většina funkcí se používá docela snadno. Nejtěžší je zřejmě vůbec nějaký jednoduchý PDF dokument vůbec vytvořit. Následující ukázka by měla pomoci začít. Vytvoří soubor `test.pdf` s jednou stránkou. Tato stránka obsahuje text "Times Roman outlined" napsaný 30ti bodovým obrysem. Text je také podtržený.

Příklad 1. Tvorba PDF dokumentu s pdflib

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = pdf_open($fp);
```

```
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php>finished</A>";
?>
```

Skript `getpdf.php` pouze vrátí vytvořený pdf dokument.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Distribuce pdflib obsahuje rozsáhlejší ukázku, která obsahuje sérii stránek s analogovými hodinami. Tato ukázka převedená do PHP vypadá takto (stejnou ukázku najdete v dokumentaci k clibpdf modulu):

Příklad 2. pdfclock ukázka z pdflib distribuce

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info($pdf, "Creator", "pdf_clock.php3");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Analog Clock");

while($pagecount- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

    /* minute strokes */
    pdf_setlinewidth($pdf, 2.0);
    for ($alpha = 0; $alpha < 360; $alpha += 6) {
        pdf_rotate($pdf, 6.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin/3, 0.0);
        pdf_stroke($pdf);
```

```
    }

    pdf_restore($pdf);
    pdf_save($pdf);

    /* 5 minute strokes */
    pdf_setlinewidth($pdf, 3.0);
    for ($alpha = 0; $alpha < 360; $alpha += 30) {
        pdf_rotate($pdf, 30.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin, 0.0);
        pdf_stroke($pdf);
    }

    $ltime = getdate();

    /* draw hour hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius/2, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw minute hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius * 0.8, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw second hand */
    pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
    pdf_setlinewidth($pdf, 2);
    pdf_save($pdf);
    pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
    pdf_moveto($pdf, -$radius/5, 0.0);
    pdf_lineto($pdf, $radius, 0.0);
    pdf_stroke($pdf);
    pdf_restore($pdf);

    /* draw little circle at center */
    pdf_circle($pdf, 0, 0, $radius/30);
    pdf_fill($pdf);

    pdf_restore($pdf);

    pdf_end_page($pdf);
}

$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php?filename=".$pdffilename.">finished</A>";
?>
```

PHP skript `getpdf.php` pouze vrátí výtvořený PDF dokument.

```
<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
```

```
fclose($fp);
?>
```

pdf_set_info (PHP 4 >= 4.0.1)

Vyplnit položku informací o dokumentu

```
void pdf_set_info (int pdf document, string fieldname, string value)
```

Funkce **pdf_set_info()** vyplní informační pole PDF dokumentu. Možné hodnoty argumentu *fieldname* jsou 'Subject', 'Title', 'Creator', 'Author', 'Keywords' a jeden uživatelsky definovaný název. Může být volána před začátkem stránky.

Příklad 1. Zadání informací o dokumentu

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Poznámka: Tato funkce nahrazuje **pdf_set_info_keywords()**, **pdf_set_info_title()**, **pdf_set_info_subject()**, **pdf_set_info_creator()** a **pdf_set_info_sybject()**.

pdf_open (PHP 3>= 3.0.6, PHP 4)

Otevřít nový PDF dokument

```
int pdf_open (int file)
```

Funkce **pdf_open()** otvírá nový PDF dokument. Odpovídající soubor musí být otevřen funkcí **fopen()** a jeho deskriptor předán jako argument *file*. Pokud této funkci nepředáte žádné argumenty, dokument bude vytvořen v paměti a vrácen stránku po stránce buď na stdout (standardní výstup) nebo do browseru.

Poznámka: Návratová hodnota je potřeba jako první argument pro všechny ostatní funkce zapisující do PDF dokumentu.

Viz také **fopen()**, **pdf_close()**.

pdf_close (PHP 3>= 3.0.6, PHP 4)

Zavřít PDF dokument

```
void pdf_close (int pdf document)
```

Funkce **pdf_close()** zavře PDF dokument.

Viz také **pdf_open()**, **fclose()**.

pdf_begin_page (PHP 3>= 3.0.6, PHP 4)

Začít novou stranu

```
void pdf_begin_page (int pdf document, double width, double height)
```

Funkce **pdf_begin_page()** začne novou stránku s výškou *height* a šířkou *width*. Pokud chcete vytvořit validní dokument, musíte zavolat tuto funkci a **pdf_end_page()** přinejmenším jednou.

Viz také **pdf_end_page()**.

pdf_end_page (PHP 3>= 3.0.6, PHP 4)

Ukončit stranu

```
void pdf_end_page (int pdf document)
```

Funkce **pdf_end_page()** ukončí stranu. Jakmile je strana ukončena, nedá se už upravovat.

Viz také **pdf_begin_page()**.

pdf_show (PHP 3>= 3.0.6, PHP 4)

Umístit text na aktuální pozici

```
void pdf_show (int pdf document, string text)
```

Funkce **pdf_show()** umístí řetězec *text* na současnou pozici s využitím aktuálního fontu.

Viz také **pdf_show_xy()**, **pdf_show_boxed()**, **pdf_set_text_pos()**, **pdf_set_font()**.

pdf_show_boxed (PHP 4 >= 4.0RC1)

Vytisknout text v rámečku

```
int pdf_show_boxed (int pdf document, string text, double x-coor, double y-coor, double
width, double height, string mode [, string feature])
```

Funkce **pdf_show_boxed()** vytiskne řetězec *text* v rámečku s levým dolním rohem na (*x-coor*, *y-coor*). Rozměry rámečku jsou *height* x *width*. Argument *mode* determines how the text is type set. Pokud jsou *width* a *height* nula, *mode* může být "left", "right" nebo "center". Pokud jsou *width* nebo *height* různé od nuly, může být také "justify" nebo "fulljustify".

Pokud má argument *feature* hodnotu "blind", text se nezobrazí.

Vrací počet znaků které nebyly zpracovány, protože se nevešly do rámečku.

Viz také **pdf_show()**, **pdf_show_xy()**.

pdf_show_xy (PHP 3>= 3.0.6, PHP 4)

Vytisknout text na určené pozici

```
void pdf_show_xy (int pdf document, string text, double x-coor, double y-coor)
```

Funkce **pdf_show_xy()** vytiskne řetězec *text* na pozici (*x-coor*, *y-coor*).

Viz také **pdf_show()**, **pdf_show_boxed()**.

pdf_set_font (PHP 3>= 3.0.6, PHP 4)

Určit font a velikost

```
void pdf_set_font (int pdf document, string font name, double size, string encoding [,
int embed])
```

Funkce **pdf_set_font()** nastaví platný font, velikost, a kódování. Pokud používáte pdflib 0.6, budete muset poskytnout Adobe Font Metrics (afm soubory) pro daný font ve font cestě (default je ./fonts). Pokud používáte PHP 3 nebo pdflib ve verzi starší než 2.20, čtvrtý argument *encoding* může mít následující hodnoty: 0 = builtin, 1 = pdfdoc, 2 = macroman, 3 = macexpert, 4 = winansi. Při *encoding* větší než 4 a menší než 0 se použije winansi. Většinou je to správná volba. Pokud používáte PHP 4 a pdflib ve verzi >= 2.20, argument *encoding* se změnil na řetězec. Používejte 'winansi', 'builtin', 'host', 'macroman' atd. Pokud má poslední argument hodnotu 1, font se vloží do PDF dokumentu. Vložit font je obvykle dobrý napad, pokud tento font není příliš rozšířený a nemůžete zaručit, že osoba, která váš dokument čte, má přístup k fontům v dokumentu použitým. Font se vloží pouze jednou, i když voláte **pdf_set_font()** několikrát.

> **Poznámka:** Pokud se má vytvořit validní PDF dokument, tato funkce se musí volat až po **pdf_begin_page()**.

> **Poznámka:** Pokud v .upr souboru odkazujete na nějaký font, ujistěte se, že jméno v afm souboru a jméno fontu jsou stejné. Jinak se font vloží vícekrát. (Díky Paulu Haddonovi za toto zjištění.)

pdf_set_leading (PHP 3>= 3.0.6, PHP 4)

Nastavit vzdálenost mezi řádky

```
void pdf_set_leading (int pdf document, double distance)
```

Funkce **pdf_set_leading()** nastavuje vzdálenost mezi řádky textu. Toto nastavení se použije, pokud se text tvoří pomocí **pdf_continue_text()**.

Viz také **pdf_continue_text()**.

pdf_set_parameter (PHP 4 >= 4.0RC1)

Nastavit určité parametry

```
void pdf_set_parameter (int pdf document, string name, string value)
```

Funkce **pdf_set_parameter()** nastaví určité parametry pdflib, které jsou typu string.

Viz také **pdf_get_value()**, **pdf_set_value()**, **pdf_get_parameter()**.

pdf_get_parameter (PHP 4 >= 4.0.1)

Zjistit hodnotu parametru

```
string pdf_get_parameter (int pdf document, string name [, double modifier])
```

Funkce **pdf_get_parameter()** function gets several parameters of pdflib which are of the type string. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0 or not passed at all.

Viz také **pdf_get_value()**, **pdf_set_value()**, **pdf_set_parameter()**.

pdf_set_value (PHP 4 >= 4.0.1)

Sets certain numerical value

```
void pdf_set_value (int pdf document, string name, double value)
```

Funkce **pdf_set_value()** function sets several numerical parameters of pdflib.

Viz také **pdf_get_value()**, **pdf_get_parameter()**, **pdf_set_parameter()**.

pdf_get_value (PHP 4 >= 4.0.1)

Gets certain numerical value

```
double pdf_get_value (int pdf document, string name [, double modifier])
```

Funkce **pdf_get_value()** function gets several numerical parameters of pdflib. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0 or not passed at all.

Viz také **pdf_set_value()**, **pdf_get_parameter()**, **pdf_set_parameter()**.

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Returns height of an image

```
string pdf_get_image_height (int pdf document, int image)
```

Funkce **pdf_get_image_height()** function returns the heights of a pdf image in pixel.

Viz také **pdf_open_image_file()**, **pdf_open_memory_image()**, **pdf_get_image_width()**.

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Returns width of an image

```
string pdf_get_image_width (int pdf document, int image)
```

Funkce **pdf_get_image_width()** function returns the widths of a pdf image in pixel.

Viz také **pdf_open_image_file()**, **pdf_open_memory_image()**, **pdf_get_image_height()**.

pdf_set_text_rendering (PHP 3>= 3.0.6, PHP 4)

Determines how text is rendered

```
void pdf_set_text_rendering (int pdf document, int mode)
```

Funkce **pdf_set_text_rendering()** function determines how text is rendered. The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to cliping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to cliping path, 7=add it to clipping path.

pdf_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4)

Sets horizontal scaling of text

```
void pdf_set_horiz_scaling (int pdf document, double scale)
```

Funkce **pdf_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

pdf_set_text_rise (PHP 3>= 3.0.6, PHP 4)

Sets the text rise

```
void pdf_set_text_rise (int pdf document, double rise)
```

Funkce **pdf_set_text_rise()** function sets the text rising to *rise* points.

pdf_set_text_matrix (PHP 3>= 3.0.6, PHP 4 <= 4.0b4)

Sets the text matrix

```
void pdf_set_text_matrix (int pdf document, array matrix)
```

Funkce **pdf_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font. The matrix has to passed as an array with six elements.

Poznámka: This function is not available anymore since pdflib 2.3

pdf_set_text_pos (PHP 3>= 3.0.6, PHP 4)

Sets text position

```
void pdf_set_text_pos (int pdf document, double x-coor, double y-coor)
```

Funkce **pdf_set_text_pos()** function sets the position of text for the next **pdf_show()** function call.

Viz také **pdf_show()**, **pdf_show_xy()**.

pdf_set_char_spacing (PHP 3>= 3.0.6, PHP 4)

Sets character spacing

```
void pdf_set_char_spacing (int pdf document, double space)
```

Funkce **pdf_set_char_spacing()** function sets the spacing between characters.

Viz také **pdf_set_word_spacing()**, **pdf_set_leading()**.

pdf_set_word_spacing (PHP 3>= 3.0.6, PHP 4)

Sets spacing between words

```
void pdf_set_word_spacing (int pdf document, double space)
```

Funkce **pdf_set_word_spacing()** function sets the spacing between words.

Viz také **pdf_set_char_spacing()**, **pdf_set_leading()**.

pdf_skew (PHP 4 >= 4.0RC1)

Skews the coordinate system

```
void pdf_skew (int pdf document, double alpha, double beta)
```

Funkce **pdf_skew()** function skew the coordinate system by *alpha* (x) and *beta* (y) degrees. *alpha* and *beta* may not be 90 or 270 degrees.

pdf_continue_text (PHP 3>= 3.0.6, PHP 4)

Outputs text in next line

```
void pdf_continue_text (int pdf document, string text)
```

Funkce **pdf_continue_text()** function outputs the string in *text* in the next line. The distance between the lines can be set with **pdf_set_leading()**.

Viz také **pdf_show_xy()**, **pdf_set_leading()**, **pdf_set_text_pos()**.

pdf_stringwidth (PHP 3>= 3.0.6, PHP 4)

Returns width of text using current font

```
double pdf_stringwidth (int pdf document, string text)
```

Funkce **pdf_stringwidth()** function returns the width of the string in *text* by using the current font. It requires a font to be set before with **pdf_set_font()**.

Viz také **pdf_set_font()**.

pdf_save (PHP 3>= 3.0.6, PHP 4)

Saves the current environment

```
void pdf_save (int pdf document)
```

Funkce **pdf_save()** function saves the current environment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects. **pdf_save()** should always be followed by **pdf_restore()** to restore the environment before **pdf_save()**.

Viz také **pdf_restore()**.

pdf_restore (PHP 3>= 3.0.6, PHP 4)

Restores formerly saved environment

```
void pdf_restore (int pdf document)
```

Funkce **pdf_restore()** function restores the environment saved with **pdf_save()**. It works like the postscript command grestore.

Příklad 1. Save and Restore

```
<?php pdf_save($pdf);
// do all kinds of rotations, transformations, ...
pdf_restore($pdf) ?>
```

Viz také **pdf_save()**.

pdf_translate (PHP 3>= 3.0.6, PHP 4)

Sets origin of coordinate system

```
void pdf_translate (int pdf document, double x-coor, double y-coor)
```

Funkce **pdf_translate()** function sets the origin of coordinate system to the point (*x-coor*, *y-coor*) relativ the current origin. The following example draws a line from (0, 0) to (200, 200) relative to the initial coordinate system. You have to set the current point after **pdf_translate()** and before you start drawing more objects.

Příklad 1. Translation

```
<?php pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
pdf_translate($pdf, 100, 100);
pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
?>
```

pdf_scale (PHP 3>= 3.0.6, PHP 4)

Sets scaling

```
void pdf_scale (int pdf document, double x-scale, double y-scale)
```

Funkce **pdf_scale()** function sets the scaling factor in both directions. The following example scales x and y direction by 72. The following line will therefore be drawn one inch in both directions.

Příklad 1. Scaling

```
<?php pdf_scale($pdf, 72.0, 72.0);
pdf_lineto($pdf, 1, 1);
pdf_stroke($pdf);
?>
```

pdf_rotate (PHP 3>= 3.0.6, PHP 4)

Sets rotation

```
void pdf_rotate (int pdf document, double angle)
```

Funkce **pdf_rotate()** function sets the rotation in degress to *angle*.

pdf_setflat (PHP 3>= 3.0.6, PHP 4)

Sets flatness

```
void pdf_setflat (int pdf document, double value)
```

Funkce **pdf_setflat()** function sets the flatness to a value between 0 and 100.

pdf_setlinejoin (PHP 3>= 3.0.6, PHP 4)

Sets linejoin parameter

```
void pdf_setlinejoin (int pdf document, long value)
```

Funkce **pdf_setlinejoin()** function sets the linejoin parameter between a value of 0 and 2.

pdf_setlinecap (PHP 3>= 3.0.6, PHP 4)

Sets linecap parameter

```
void pdf_setlinecap (int pdf document, int value)
```

Funkce **pdf_setlinecap()** function sets the linecap parameter between a value of 0 and 2.

pdf_setmiterlimit (PHP 3>= 3.0.6, PHP 4)

Sets miter limit

```
void pdf_setmiterlimit (int pdf document, double value)
```

Funkce **pdf_setmiterlimit()** function sets the miter limit to a value greater of equal than 1.

pdf_setlinewidth (PHP 3>= 3.0.6, PHP 4)

Sets line width

```
void pdf_setlinewidth (int pdf document, double width)
```

Funkce **pdf_setlinewidth()** function sets the line width to *width*.

pdf_setdash (PHP 3>= 3.0.6, PHP 4)

Sets dash pattern

```
void pdf_setdash (int pdf document, double white, double black)
```

Funkce **pdf_setdash()** function sets the dash pattern *white* white points and *black* black points. If both are 0 a solid line is set.

pdf_moveto (PHP 3>= 3.0.6, PHP 4)

Sets current point

```
void pdf_moveto (int pdf document, double x-coor, double y-coor)
```

Funkce **pdf_moveto()** function sets the current point to the coordinates *x-coor* and *y-coor*.

pdf_curveto (PHP 3>= 3.0.6, PHP 4)

Draws a curve

```
void pdf_curveto (int pdf document, double x1, double y1, double x2, double y2, double
x3, double y3)
```

Funkce **pdf_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

Viz také **pdf_moveto()**, **pdf_lineto()**, **pdf_stroke()**.

pdf_lineto (PHP 3>= 3.0.6, PHP 4)

Draws a line

```
void pdf_lineto (int pdf document, double x-coor, double y-coor)
```

Funkce **pdf_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

Viz také **pdf_moveto()**, **pdf_curveto()**, **pdf_stroke()**.

pdf_circle (PHP 3>= 3.0.6, PHP 4)

Draws a circle

```
void pdf_circle (int pdf document, double x-coor, double y-coor, double radius)
```

Funkce **pdf_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

Viz také **pdf_arc()**, **pdf_stroke()**.

pdf_arc (PHP 3>= 3.0.6, PHP 4)

Draws an arc

```
void pdf_arc (int pdf document, double x-coor, double y-coor, double radius, double
start, double end)
```

Funkce **pdf_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

Viz také **pdf_circle()**, **pdf_stroke()**.

pdf_rect (PHP 3>= 3.0.6, PHP 4)

Draws a rectangle

```
void pdf_rect (int pdf document, double x-coor, double y-coor, double width, double
height)
```

Funkce **pdf_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *width*. This height is set to *height*.

Viz také **pdf_stroke()**.

pdf_closepath (PHP 3>= 3.0.6, PHP 4)

Closes path

```
void pdf_closepath (int pdf document)
```

Funkce **pdf_closepath()** function closes the current path. This means, it draws a line from current point to the point where the first line was started. Many functions like **pdf_moveto()**, **pdf_circle()** and **pdf_rect()** start a new path.

pdf_stroke (PHP 3>= 3.0.6, PHP 4)

Draws line along path

```
void pdf_stroke (int pdf document)
```

Funkce **pdf_stroke()** function draws a line along current path. The current path is the sum of all line drawing. Without this function the line would not be drawn.

Viz také **pdf_closepath()**, **pdf_closepath_stroke()**.

pdf_closepath_stroke (PHP 3>= 3.0.6, PHP 4)

Closes path and draws line along path

```
void pdf_closepath_stroke (int pdf document)
```

Funkce **pdf_closepath_stroke()** function is a combination of **pdf_closepath()** and **pdf_stroke()**. It also clears the path.

Viz také **pdf_closepath()**, **pdf_stroke()**.

pdf_fill (PHP 3>= 3.0.6, PHP 4)

Fills current path

```
void pdf_fill (int pdf document)
```

Funkce **pdf_fill()** function fills the interior of the current path with the current fill color.

Viz také **pdf_closepath()**, **pdf_stroke()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Fills and strokes current path

```
void pdf_fill_stroke (int pdf document)
```

Funkce **pdf_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

Viz také **pdf_closepath()**, **pdf_stroke()**, **pdf_fill()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Closes, fills and strokes current path

```
void pdf_closepath_fill_stroke (int pdf document)
```

Funkce **pdf_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

Viz také **pdf_closepath()**, **pdf_stroke()**, **pdf_fill()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_endpath (PHP 3>= 3.0.6, PHP 4)

Ends current path

```
void pdf_endpath (int pdf document)
```

Funkce **pdf_endpath()** function ends the current path but does not close it.

Viz také **pdf_closepath()**.

pdf_clip (PHP 3>= 3.0.6, PHP 4)

Clips to current path

```
void pdf_clip (int pdf document)
```

Funkce **pdf_clip()** function clips all drawing to the current path.

pdf_setgray_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to gray value

```
void pdf_setgray_fill (int pdf document, double gray value)
```

Funkce **pdf_setgray_fill()** function sets the current gray value to fill a path.

Viz také **pdf_setrgbcolor_fill()**.

pdf_setgray_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to gray value

```
void pdf_setgray_stroke (int pdf document, double gray value)
```

Funkce **pdf_setgray_stroke()** function sets the current drawing color to the given gray value.

Viz také **pdf_setrgbcolor_stroke()**.

pdf_setgray (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to gray value

```
void pdf_setgray (int pdf document, double gray value)
```

Funkce **pdf_setgray()** function sets the current drawing and filling color to the given gray value.

Viz také **pdf_setrgbcolor_stroke()**, **pdf_setrgbcolor_fill()**.

pdf_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to rgb color value

```
void pdf_setrgbcolor_fill (int pdf document, double red value, double green value,
double blue value)
```

Funkce **pdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

Viz také **pdf_setrgbcolor_fill()**.

pdf_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to rgb color value

```
void pdf_setrgbcolor_stroke (int pdf document, double red value, double green value,
double blue value)
```

Funkce **pdf_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

Viz také **pdf_setrgbcolor_stroke()**.

pdf_setrgbcolor (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to rgb color value

```
void pdf_setrgbcolor (int pdf document, double red value, double green value, double
blue value)
```

Funkce **pdf_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

Viz také **pdf_setrgbcolor_stroke()**, **pdf_setrgbcolor_fill()**.

pdf_add_outline (PHP 3>= 3.0.6, PHP 4)

Adds bookmark for current page

```
int pdf_add_outline (int pdf document, string text [, int parent [, int open]])
```

Funkce **pdf_add_outline()** function adds a bookmark with text *text* that points to the current page. The bookmark is inserted as a child of *parent* and is by default open if *open* is not 0. The return value is an identifier for the bookmark which can be used as a parent for other bookmarks. Therefore you can build up hierarchies of bookmarks.

Unfortunately pdflib does not make a copy of the string, which forces PHP to allocate the memory. Currently this piece of memory is not been freed by any PDF function but it will be taken care of by the PHP memory manager.

pdf_set_transition (PHP 3>= 3.0.6, PHP 4)

Sets transition between pages

```
void pdf_set_transition (int pdf document, int transition)
```

Funkce **pdf_set_transition()** function set the transition between following pages. The value of *transition* can be

0 for none,

1 for two lines sweeping across the screen reveal the page,

2 for multiple lines sweeping across the screen reveal the page,

3 for a box reveals the page,

4 for a single line sweeping across the screen reveals the page,
5 for the old page dissolves to reveal the page,
6 for the dissolve effect moves from one screen edge to another,
7 for the old page is simply replaced by the new page (default)

Viz také **pdf_set_duration()**.

pdf_set_duration (PHP 3>= 3.0.6, PHP 4)

Sets duration between pages

```
void pdf_set_duration (int pdf document, double duration)
```

Funkce **pdf_set_duration()** function set the duration between following pages in seconds.

Viz také **pdf_set_transition()**.

pdf_open_gif (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Opens a GIF image

```
int pdf_open_gif (int pdf document, string filename)
```

Funkce **pdf_open_gif()** function opens an image stored in the file with the name *filename*. The format of the image has to be gif. The function returns a pdf image identifier.

Poznámka: This function shouldn't be used anymore. Please use the function **pdf_open_image_file()** instead.

Příklad 1. Including a gif image

```
<?php
$im = pdf_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

Viz také **pdf_close_image()**, **pdf_open_jpeg()**, **pdf_open_memory_image()**, **pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_open_png (PHP 4 >= 4.0RC2)

Opens a PNG image

```
int pdf_open_png  (int pdf, string png_file)
```

Funkce **pdf_open_png()** function opens an image stored in the file with the name *filename*. The format of the image has to be png. The function returns a pdf image identifier.

Poznámka: This function shouldn't be used anymore. Please use the function **pdf_open_image_file()** instead.

Příklad 1. Including a PNG image

```
<?php
$im = pdf_open_png ($pdf, "test.png");
pdf_place_image ($pdf, $im, 100, 100, 1);
pdf_close_image ($pdf, $im);
?>
```

Viz také **pdf_close_image()**, **pdf_open_jpeg()**, **pdf_open_gif()**, **pdf_open_memory_image()**, **pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_open_image_file (PHP 3 CVS only, PHP 4 >= 4.0RC2)

Reads an image from a file

```
int pdf_open_image_file (int PDF-document, string format, string filename)
```

The function **pdf_open_image_file()** reads an image of format *format* from the file *filename*. Possible formats are 'png', 'tiff', 'jpeg' and 'gif'. The function returns a pdf image identifier.

Příklad 1. Inserting an image

```
<?php
$pim = pdf_open_image_file($pdf, "png", "picture.png");
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

Viz také **pdf_close_image()**, **pdf_open_jpeg()**, **pdf_open_gif()**, **pdf_open_tiff()**, **pdf_open_png()**, **pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_open_memory_image (PHP 3>= 3.0.10, PHP 4 >= 4.0b2)

Opens an image created with PHP's image functions

```
int pdf_open_memory_image (int pdf document, int image)
```

Funkce **pdf_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the PDF dokument. The function returns a pdf image identifier.

Příklad 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

Viz také **pdf_close_image()**, **pdf_open_jpeg()**, **pdf_open_gif()**, **pdf_open_png() pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_open_jpeg (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Opens a JPEG image

```
int pdf_open_jpeg (int pdf document, string filename)
```

Funkce **pdf_open_jpeg()** function opens an image stored in the file with the name *filename*. The format of the image has to be jpeg. The function returns a pdf image identifier.

Poznámka: This function shouldn't be used anymore. Please use the function **pdf_open_image_file()** instead.

Viz také **pdf_close_image()**, **pdf_open_gif()**, **pdf_open_png()**, **pdf_open_memory_image()**, **pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_open_tiff (PHP 4 >= 4.0b4)

Opens a TIFF image

```
int pdf_open_tiff (int PDF-document, string filename)
```

Funkce **pdf_open_tiff()** function opens an image stored in the file with the name *filename*. The format of the image has to be tiff. The function returns a pdf image identifier.

Poznámka: This function shouldn't be used anymore. Please use the function **pdf_open_image_file()** instead.

Viz také **pdf_close_image()**, **pdf_open_gif()**, **pdf_open_jpeg()**, **pdf_open_png()**, **pdf_open_memory_image()**, **pdf_execute_image()**, **pdf_place_image()**, **pdf_put_image()**.

pdf_close_image (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Closes an image

```
void pdf_close_image (int image)
```

Funkce **pdf_close_image()** function closes an image which has been opened with any of the **pdf_open_xxx()** functions.

Viz také **pdf_open_jpeg()**, **pdf_open_gif()**, **pdf_open_memory_image()**.

pdf_place_image (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Places an image on the page

```
void pdf_place_image (int pdf document, int image, double x-coor, double y-coor, double scale)
```

Funkce **pdf_place_image()** function places an image on the page at postion (*x-coor*, *x-coor*). The image can be scaled at the same time.

Viz také **pdf_put_image()**.

pdf_put_image (PHP 3>= 3.0.7, 4.0b2 - 4.0b4 only)

Stores an image in the PDF for later use

```
void pdf_put_image (int pdf document, int image)
```

Funkce **pdf_put_image()** function places an image in the PDF file without showing it. The stored image can be displayed with the **pdf_execute_image()** function as many times as needed. This is useful when using the same image multiple times in order to keep the file size small. Using **pdf_put_image()** and **pdf_execute_image()** is highly recommended for larger images (several kb) if they show up more than once in the document.

Poznámka: This function has become meaningless with version 2.01 of pdflib. It will just output a warning.

Viz také **pdf_put_image()**, **pdf_place_image()**, **pdf_execute_image()**.

pdf_execute_image (PHP 3>= 3.0.7, 4.0b2 - 4.0b4 only)

Places a stored image on the page

```
void pdf_execute_image (int pdf document, int image, double x-coor, double y-coor,
double scale)
```

Funkce **pdf_execute_image()** function displays an image that has been put in the PDF file with the **pdf_put_image()** function on the current page at the given coordinates.

The image can be scaled while displaying it. A scale of 1.0 will show the image in the original size.

Poznámka: This function has become meaningless with version 2.01 of pdflib. It will just output a warning.

Příklad 1. Multiple show of an image

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = pdf_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2);
pdf_close_image($pdf, $pim);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Adds annotation

```
void pdf_add_annotation (int pdf document, double llx, double lly, double urx, double
ury, string title, string content)
```

Funkce **pdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

pdf_set_border_style (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Sets style of border around links and annotations

```
void pdf_set_border_style (int pdf document, string style, double width)
```

Funkce **pdf_set_border_style()** function sets the style and width of the suroundig box of links and annotations. Argument*style* can be 'solid' or 'dashed'.

Viz také **pdf_set_border_color()**, **pdf_set_border_dash()**.

pdf_set_border_color (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Sets color of border around links and annotations

```
void pdf_set_border_color (int pdf document, double red, double green, double blue)
```

Funkce **pdf_set_border_color()** function sets the color of the suroundig box of links and annotations. The three color components have to have a value between 0.0 and 1.0.

Viz také **pdf_set_border_style()**, **pdf_set_border_dash()**.

pdf_set_border_dash (PHP 4 >= 4.0.1)

Sets dash style of border around links and annotations

```
void pdf_set_border_dash (int pdf document, double black, double white)
```

Funkce **pdf_set_border_dash()** function sets the lenght of black and white areas of a dashed line of the suroundig box of links and annotations.

Viz také **pdf_set_border_style()**, **pdf_set_border_color()**.

LVII. Funkce pro práci s Verisign Payflow Pro

Tato extenze umožňuje zpracovávat kreditní karty a provádět jiné finanční transakce pomocí Verisign Payment Services (dříve Signio, http://www.verisign.com/payment/).

Tyto funkce jsou dostupné pouze pokud bylo PHP zkompilováno s `-with-pfpro[=DIR]`. Budete potřebovat SDK pro vaši platformu, který se dá po registraci stáhnout z manažerského rozhraní (https://testmanager.signio.com/Downloads/Downloads_secure.htm).

Pokud jste si stáhli správný SDK, zkopírujte následující soubory z `lib` adresáře této distribuce: `pfpro.h` do `/usr/local/include` a `libpfpro.so` do `/usr/local/lib`.

Při využívání těchto funkcí můžete vynechat volání **pfpro_init()** a **pfpro_cleanup()**, tato extenze to udělá podle potřeby automaticky. Tyto funkce jsou ale přesto dostupné pro případ, že byste potřebovali zpracovávat velké množství transakcí a vyžadovali naprostou kontrolu nad touto knihovnou. Mezi **pfpro_init()** a **pfpro_cleanup()** můžete provést libovolné množství transakcí.

Tyto funkce byly přidány v PHP 4.0.2.

> **Poznámka:** Tyto funkce poskytují pouze spojení s Verisign Payment Services. Kompletní detaily vyžadovaných parametrů viz Payflow Pro Developer's Guide.

pfpro_init (PHP 4 >= 4.0.2)

Inicializovat Payflow Pro knihovnu

```
void pfpro_init(void);
```

pfpro_init() se používá k inicializaci Payflow Pro knihovny. Tuto funkci volat nemusíte, tato extenze automaticky zavolá **pfpro_init()** před první transakcí.

Viz také **pfpro_cleanup()**.

pfpro_cleanup (PHP 4 >= 4.0.2)

Zavřít Payflow Pro knihovnu

```
void pfpro_cleanup(void);
```

pfpro_cleanup() se používá k čistému vypnutí Payflow Pro knihovny. Měla by se volat po provedení všech transakcí a před ukončením skriptu. Tuto funkci nicméně volat nemusíte, tato extenze automaticky zavolá **pfpro_cleanup()** při ukončení skriptu.

Viz také **pfpro_init()**.

pfpro_process (PHP 4 >= 4.0.2)

Zpracovat transakci s Payflow Pro

```
array pfpro_process (array parameters [, string address [, int port [, int timeout [,
string proxy address [, int proxy port [, string proxy logon [, string proxy
password]]]]]]])
```

Vrací asociativní pole obsahující odpověď.

pfpro_process() zpracuje transakci s Payflow Pro. První argument je asociativní pole obsahující klíče a hodnoty, které se zakódují a odešlou zpracovateli.

Druhý argument je volitelný a určuje serveer, ke kterému se připojit. Default je "test.signio.com", takže pokud chcete zpracovávat skutečné transakce, budete chtít tento argument nastavit na "connect.signio.com".

Třetí argument určuje port, ke kterému se připojit. Default je 443, standardní SSL port.

Čtvrtý argument určuje v sekundách, jaký časový limit se má použít. Default je 30 sekund. Tento časový limit vstupuje v platnost v okamžiku spojení se zpracovatelem, a tak by váš skript mohl potenciálně běžet velmi dlouhou dobu, pokud by nastaly problémy s DNS nebo sítí.

Pátý argument určuje hostname vaší případné SSL proxy. Šestý argument specifikuje port.

Sedmý a osmý argument určují přihlašovací jméno a heslo na tuto proxy.

Tato funkce vrací asociativní pole klíčů a hodnot odpovědi.

> **Poznámka:** Kompletní detaily vyžadovaných parametrů viz Payflow Pro Developer's Guide.

Příklad 1. Ukázka Payflow Pro

```
<?php

pfpro_init();
```

```
$transaction = array(USER => 'login',
     PWD => 'heslo',
     TRXTYPE => 'S',
     TENDER => 'C',
     AMT => 1.50,
     ACCT => '4111111111111111',
     EXPDATE => '0904'
     );

$response = pfpro_process($transaction);

if (!$response) {
  die("Nepodařilo se spojit s Verisign.\n");
}

echo "Response kód Verisignu byl ".$response[RESULT];
echo ", což znamená: ".$response[RESPMSG]."\n";

echo "\nPožadavek na transakci: ";
print_r($transaction);

echo "\nOdpověď': ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_process_raw (PHP 4 >= 4.0.2)

Zpracovat raw transakci s Payflow Pro

```
string pfpro_process_raw (string parameters [, string address [, int port [, int timeout
[, string proxy address [, int proxy port [, string proxy logon [, string proxy
password]]]]]]])
```

Vrací řetězec obsahující odpověď.

pfpro_process_raw() zpracuje raw řetězec transakce s Payflow Pro. Opravdu byste ale měli používat **pfpro_process()**, protože pravidla kódování těchto transakcí jsou nestandardní.

První argument je v tomto případě řetězec obsahující raw požadavek na transakci. Všechny ostatní argumenty jsou stejné jako u **pfpro_process()**. Návratová hodnota je řetězec obsahující raw odpověď.

> **Poznámka:** Kompletní detaily vyžadovaných parametrů a pravidel kódování viz Payflow Pro Developer's Guide. Dobře vám radíme, používejte radši **pfpro_process()**.

Příklad 1. Ukázka Payflow Pro raw

```
<?php

pfpro_init();

$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=411111111

if (!$response) {
  die("Nepodařilo se spojit s Verisign.\n");
}

echo "Raw odpověď Verisignu byla ".$response;
```

```
pfpro_cleanup();

?>
```

pfpro_version (PHP 4 >= 4.0.2)

Vrátit verzi Payflow Pro knihovny

```
string pfpro_version(void);
```

pfpro_version() vrací řetězec obsahující verzi Payflow Pro knihovny. V čase psaní tohoto manuálu to bylo L211.

LVIII. PHP volby & informace

assert (PHP 4 >= 4.0b4)

Ověřit, jestli je výrok neplatný

```
int assert (string|bool assertion)
```

assert() ověří předanou *assertion* a provede příslušnou akci, pokud je výsledek `false`.

Pokud je předaná *assertion* řetězec, vyhodnotí se funkcí **assert()** jako PHP kód. Výhody řetězcové *assertion* jsou menší režie, když je kontrola výroků vypnutá, a zprávy obsahující *assertion* výraz, když výrok selže.

Kontrola výroků by se měla používat jen pro odlaďování skriptů. Můžete je použít na kontrolu podmínek, které by měly být vždycky true, a které jinak indikují nějaké chyby v programování, nebo na kontrolu existence určitých vlastností, jako jsou funkce obsažené v extenzích, nebo určité systémové limity a vlastnosti.

Výroky by se neměly používat pro běžné operace jako kontrola vstupních parametrů. Jako základní pravidlo platí, že váš kód by měl fungovat správně, pokud není kontrola výroků aktivována.

Chování funkce **assert()** lze konfigurovat skrze **assert_options()** nebo .ini direktivy popsané na manuálové stránce této funkce.

assert_options (PHP 4 >= 4.0b4)

Nastavit/získat různé příznaky výroků

```
mixed assert_options (int what [, mixed value])
```

S použitím funkce **assert_options()** můžete nastavit různé řídící volby funkce **assert()**, nebo jen získat jejich současné nastavení.

Tabulka 1. Volby výroků

volba	ini parametr	výchozí hodnota	popis
ASSERT_ACTIVE	assert.active	1	zapne **assert()** vyhodnocování
ASSERT_WARNING	assert.warning	1	vytvoří PHP varování pro každý selhavší výrok
ASSERT_BAIL	assert.bail	0	ukončí provádění skiptu, pokud výrok selže
ASSERT_QUIET_EVAL	assert.quiet_eval	0	vypne error_reporting během vyhodnocování výrazů výroků
ASSERT_CALLBACK	assert_callback	(null)	uživatelská funkce, která se zavolá pro každý selhavší výrok

assert_options() vrací původní nastavení volby, nebo `false` při chybě.

extension_loaded (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

zjistit, jestli je extenze načtená

```
bool extension_loaded (string name)
```

Vrací true, pokud je extenze identifikovaná argumentem *name* načtena. Názvy různých extenzí můžete vidět, pokud použijete **phpinfo()**.

Viz také **phpinfo()**.

Poznámka: Tato funkce byla přidána v PHP 3.0.10.

dl (PHP 3, PHP 4)

Načíst extenzi PHP za běhu

```
int dl (string library)
```

Načte extenzi PHP definovanou v *library*. Viz také konfigurační direktivu extension_dir.

getenv (PHP 3, PHP 4)

Získat hodnotu systémové proměnné

```
string getenv (string varname)
```

Vrátí hodnotu systémové proměnné *varname*, nebo false při chybě.

```
$ip = getenv ("REMOTE_ADDR"); // získá IP adresu uživatele
```

Seznam všech systémových proměnných si můžete prohlédnout použitím **phpinfo()**. Význam mnoha z nich najdete v CGI specifikaci (http://hoohoo.ncsa.uiuc.edu/cgi/), zvláště na stránce o systémových proměnných (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

Poznámka: Tato funkce nefunguje v ISAPI módu.

get_cfg_var (PHP 3, PHP 4)

Získat hodnotu volby z konfigurace PHP

```
string get_cfg_var (string varname)
```

Vrátí současnou hodnotu konfigurační proměnné PHP určené argumentem *varname*, nebo false pokud dojde k chybě.

Nevrací konfigurační hodnoty nastavené při kompilaci PHP a nečte konfigurační soubor Apache (použití php3_configuration_option direktiv).

Pokud chcete zjistit, jestli daný systém používá konfigurační soubor, zkuste získat hodnotu konfigurační volby cfg_file_path. Pokud je tato dostupná, používá se konfigurační soubor.

get_current_user (PHP 3, PHP 4)

Získat jméno vlastníka současného PHP skriptu

```
string get_current_user (void)
```

Vrátí jméno vlastníka současného PHP skriptu.

Viz také **getmyuid()**, **getmypid()**, **getmyinode()**, a **getlastmod()**.

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4)

Získat současné aktivní nastavení magic_quotes_gpc

```
long get_magic_quotes_gpc (void)
```

Vrátí současné aktivní nastavení magic_quotes_gpc. (0 pro vypnuto, 1 pro zapnuto).

Viz také **get_magic_quotes_runtime()**, **set_magic_quotes_runtime()**.

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Vrátit současné aktivní nastavení magic_quotes_runtime

```
long get_magic_quotes_runtime  (void)
```

Vrátí současné aktivní nastavení magic_quotes_runtime. (0 pro vypnuto, 1 pro zapnuto).

Viz také **get_magic_quotes_gpc()**, **set_magic_quotes_runtime()**.

getlastmod (PHP 3, PHP 4)

Získat čas poslení modifikace skriptu

```
int getlastmod (void)
```

Vrátí čas poslení modifikace současné stránky. Návratová hodnota je Unixový timestamp, vhodný jako vstup pro **date()**. Při chybě vrací false.

Příklad 1. getlastmod() příklad

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date ("F d Y H:i:s.", getlastmod());
```

See alse **date()**, **getmyuid()**, **get_current_user()**, **getmyinode()**, and **getmypid()**.

getmyinode (PHP 3, PHP 4)

Získat inode současného skriptu

```
int getmyinode (void)
```

Vrátí inode současného skriptu, nebo false při chybě.

Viz také **getmyuid()**, **get_current_user()**, **getmypid()**, and **getlastmod()**.

Poznámka: Tato funkce není podporována na Windows systémech.

getmypid (PHP 3, PHP 4)

Získat process ID PHP

```
int getmypid (void)
```

Vrátí process ID současného PHP procesu, nebo false při chybě.

> **Varování**
>
> Process ID nejsou unikátní, a jsou tudíž slabým zdrojem entropie. Nedoporučujeme používat PIDy v prostředích závislých na bezpečnosti.

Viz také **getmyuid()**, **get_current_user()**, **getmyinode()**, a **getlastmod()**.

getmyuid (PHP 3, PHP 4)

Získat UID majitele PHP skriptu

```
int getmyuid (void)
```

Vrátí user ID současného skriptu, nebo false při chybě.

Viz také **getmypid()**, **get_current_user()**, **getmyinode()**, a **getlastmod()**.

getrusage (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Získat informace o současném využití zdrojů

```
array getrusage ([int who])
```

Toto je rozhraní ke to getrusage(2). Vrátí asociativní pole obsahující všechna data vrácená systémovým voláním. Pokud je who 1, getrusage se zavolá s RUSAGE_CHILDREN.

Všechny položky jsou přístupné skrze svá dokumentovaná jména.

Příklad 1. Getrusage příklad

```
$dat = getrusage();
echo $dat["ru_nswap"];         # počet swapů
echo $dat["ru_majflt"];        # počet page faultů
echo $dat["ru_utime.tv_sec"];  # využitý uživatelský čas (sekundy)
echo $dat["ru_utime.tv_usec"]; # využitý uživatelský čas (mikrosekundy)
```

Další detaily viz man stránka getrusage(2) na vašem systému.

ini_alter (PHP 4)

Změnit hodnotu konfigurační volby

```
string ini_alter (string varname, string newvalue)
```

Změní hodnotu konfigurační volby, vrátí `false` při selhání, a předchozí hodnotu konfigurační volby při úspěchu.

> **Poznámka:** Toto je alias k **ini_set()**

Viz také **ini_get()**, **ini_restore()**, **ini_set()**

ini_get (PHP 4)

Získat hodnotu konfigurační volby

```
string ini_get (string varname)
```

Vrátí hodnotu konfigurační volby při úspěchu, `false` při selhání.

Viz také **ini_alter()**, **ini_restore()**, **ini_set()**

ini_restore (PHP 4)

Obnovit původní hodnotu konfigurační volby

```
string ini_restore (string varname)
```

Obnoví původní hodnotu konfigurační volby.

Viz také **ini_alter()**, **ini_get()**, **ini_set()**

ini_set (PHP 4 >= 4.0RC1)

Změnit hodnotu konfigurační volby

```
string ini_set (string varname, string newvalue)
```

Změní hodnotu konfigurační volby, vrátí `false` při selhání, a předchozí hodnotu konfigurační volby při úspěchu.

Viz také **ini_alter()**, **ini_get()**, **ini_restore()**

phpcredits (PHP 4)

Vytisknout credits pro PHP

```
void phpcredits (int flag)
```

Tato funkce vytiskne credits vč. seznamu vývojářů PHP, modulů, atd. Generuje příslušný HTML kód, kterým se tyto informace vkládají do stránky. Je třeba předat argument indikující co se vytiskne (předdefinovaná konstanta flag, viz níže). Například k vytištění všeobecných credits můžete někde ve svém kódu použít:

```
...
phpcredits(CREDITS_GENERAL);
...
```

A pokud chcete vytisknout užší kruh vývojářů a dokumentační skupinu na samostatné stránce, použijte:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

A pokud chcete vložit všechny credits do vlastní stránky, pomůže vám následující kód:

```
<html>
 <head>
  <title>Má stránka s credits</title>
 </head>
 <body>
  <?php
// váš vlastní kód
phpcredits(CREDITS_ALL + CREDITS_FULLPAGE);
// další kód
?>
 </body>
</html>
```

Tabulka 1. Předdefinované phpcredits() příznaky

název	popis
CREDITS_ALL	Všechny credits, ekvivalentní k: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. Vygeneruje kompletní samostatnou HTML stránku s příslušnými tagy.
CREDITS_DOCS	Credits dokumentačního týmu
CREDITS_FULLPAGE	Obvykle se používá v kombinaci s jinými příznaky. Indikuje, že se má vytisknout kompletní samostatná HTML stránka, včetně informací indikovaných jinými příznaky.
CREDITS_GENERAL	Všeobecné credits: Design a koncept jazyka, autoři PHP 4.0 a SAPI modul.
CREDITS_GROUP	Seznam užšího kruhu vývojářů
CREDITS_MODULES	Seznam rozšiřujících modulů (extenzí) PHP a jejich autorů
CREDITS_SAPI	Seznam server API modulů PHP a jejich autorů

Viz také **phpinfo()**, **phpversion()**, **php_logo_guid()**.

phpinfo (PHP 3, PHP 4)

Vytisknout spoustu informací o PHP

```
int phpinfo ([int what])
```

Vytiskne velké množství informací o současném stavu PHP. To zahrnuje informace o kompilačních volbách a extenzích, verzi PHP, informaci o serveru a systémů (pokud je PHP zkompilováno jako modul), PHP prostředí, verzi OS, cesty, hlavní a lokální hodnoty konfiguračních voleb, HTTP hlavičky, a PHP licenci.

Výstup se dá upravit předáním jednou nebo více z následujících hodnot uložených ve volitelném argumentu *what*.

- INFO_GENERAL
- INFO_CREDITS
- INFO_CONFIGURATION
- INFO_MODULES
- INFO_ENVIRONMENT
- INFO_VARIABLES
- INFO_LICENSE
- INFO_ALL

Viz také **phpversion()**, **phpcredits()**, **php_logo_guid()**

phpversion (PHP 3, PHP 4)

Získat současnou verzi PHP

```
string phpversion (void)
```

Vrátí řetězec obsahující verzi právě běžícího PHP parseru.

Příklad 1. phpversion() example

```
// prints e.g. 'Současná verze PHP: 3.0rel-dev'
echo "Současná verze PHP: ".phpversion();
```

Viz také **phpinfo()**, **phpcredits()**, **php_logo_guid()**

php_logo_guid (PHP 4 >= 4.0b4)

Get the logo guid

```
string php_logo_guid (void)
```

Poznámka: Tato funkcionalita byla přidána v PHP 4 Beta 4.

Viz také **phpinfo()**. **phpversion()**, **phpcredits()**

php_sapi_name (PHP 4 >= 4.0.1)

Vrátit typ rozhraní mezi web serverem a PHP Returns the type of interface between web server and PHP

```
string php_sapi_name(void);
```

php_sapi_name() vrátí řetězec popisující malými písmeny typ rozhraní mezi web serverem a PHP (Server API, SAPI). U CGI PHP je tento řetězec "cgi", u mod_php pro Apache je tento řetězec "apache" a tak dále.

Příklad 1. Php_sapi_name() Example

```
$sapi_type = php_sapi_name();
if ($sapi_type == "cgi")
  print "Používáte CGI PHP\n";
else
  print "Nepoužíváte CGI PHP\n";
```

php_uname (PHP 4 >= 4.0.2)

Vrátit informaci o operačním systému, na kterém bylo PHP zkompilováno

```
string php_uname(void);
```

php_uname() vrátí řetězec s popisem operačního systému, na kterém bylo PHP zkompilováno.

Příklad 1. php_uname() Example

```
if (substr(php_uname(), 0, 7) == "Windows") {
  die("Pardon, tento skript na Windows nefunguje.\n");
}
```

putenv (PHP 3, PHP 4)

Nastavit hodnotu systémové proměnné

```
void putenv (string setting)
```

Přidá *setting* do prostředí serveru.

Příklad 1. Nastavení systémové proměnné

```
putenv ("UNIQID=$uniqid");
```

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Nastavit současnou aktivní hodnotu magic_quotes_runtime

```
long set_magic_quotes_runtime  (int new_setting)
```

Nastaví současnou aktivní konfigurační volby magic_quotes_runtime. (0 vypnuto, 1 zapnuto)

Viz také **get_magic_quotes_gpc()**, **get_magic_quotes_runtime()**.

set_time_limit (PHP 3, PHP 4)

Omezit maximální dobu průběhu skriptu

```
void set_time_limit (int seconds)
```

Určí počet sekund po které může skript běžet. Pokud je dosaženo tohoto času, skript vrátí fatální chybu. Standardní limit je 30 sekund, nebo, pokud existuje, hodnota direktivy max_execution_time definovaná v konfiguračním souboru. Pokud je *seconds* nula, neexistuje žádný časový limit.

set_time_limit() při svém zavolání restartuje čítač času od nuly. Jinými slovy, pokud je limit standardních šé sekund a po 25 sekundách provádění skriptu dojde k volání set_time_limit(20), tento skript poběží celkem 45 sekund předtím, než skončí na časovém limitu.

Všimněte si, že **set_time_limit()** nemá žádný účinek, když PHP běží v bezpečném módu. Obejít to lz jedine vypnutím bezpečného módu nebo změnou časového limitu v konfiguračním souboru.

zend_logo_guid (PHP 4 >= 4.0b4)

Získat zend guid

```
string zend_logo_guid (void)
```

Poznámka: Tato funkcionalita byla přidána v PHP 4 Beta 4.

get_loaded_extensions (PHP 4 >= 4.0b4)

Vrátit pole se jmény všech zkompilovaných a načtených modulů (extenzí)

```
array get_loaded_extensions (void)
```

Tato funkce vrací jména všech modulů (extenzí) zkompilovaných a načtených do PHP interpretru.

Například následující řádek

```
print_r (get_loaded_extensions());
```

vypíše seznam podobný tomuto:

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
  [3] => session
  [4] => posix
  [5] => pgsql
  [6] => pcre
  [7] => gd
  [8] => ftp
  [9] => db
  [10] => Calendar
  [11] => bcmath
)
```

Viz také: **get_extension_funcs()**.

get_extension_funcs (PHP 4 >= 4.0b4)

Vrátit pole jmen funkcí určitého modulu

```
array get_extension_funcs (string module_name)
```

Tato funkce vrací jména všech funkcí definovaných v modulu určeném argumentem *module_name*.

Například následující řádky

```
print_r (get_extension_funcs ("xml"));
print_r (get_extension_funcs ("gd"));
```

vytisknou seznam všech funkcí v modulech `xml` a `gd`.

Viz také: **get_loaded_extensions()**

get_required_files (PHP 4 >= 4.0RC2)

Vrátit pole jmen všech souborů, které byly v určitém skriptu načteny pomocí require_once()

```
array get_required_files (void)
```

Tato funkce vrátí asociativní pole jmen všech souborů, které byly načteny do probíhajícího skriptu pomocí **require_once()**. Indexy tohoto pole jsou názvy souborů použitých v **require_once()** bez přípony ".php".

Následující příklad

Příklad 1. Tisk require()ovaných a include()ovaných souborů

```
<?php

require_once ("local.php");
require_once ("../inc/global.php");

for ($i=1; $i<5; $i++)
  include "util".$i."php";

  echo "Required_once files\n";
  print_r (get_required_files());

  echo "Included_once files\n";
  print_r (get_included_files());
?>
```

vygeneruje následující výstup:

```
Required_once files
Array
(
  [local] => local.php
  [../inc/global] => /full/path/to/inc/global.php
)

Included_once files
Array
(
    [util1] => util1.php
    [util2] => util2.php
    [util3] => util3.php
    [util4] => util4.php
)
```

Poznámka: Od verze PHP 4.0.1pl2 tato funkce předpokládá, že soubory v `required_once` končí příponou ".php", jiné přípony nefungují.

Viz také: **require_once()**, **include_once()**, **get_included_files()**

get_included_files (PHP 4 >= 4.0RC1)

Vrátit pole jmen všech souborů, které byly ve skriptu načteny pomocí include_once()

```
array get_included_files (void)
```

Tato funkce vrátí asociativní pole jmen všech souborů, které byly načteny do skriptu pomocí **include_once()**. Indexy tohoto pole jsou názvy souborů použitých v **include_once()** bez přípony ".php".

Poznámka: Od verze PHP 4.0.1pl2 tato funkce předpokládá, že soubory v `include_once` končí příponou ".php", jiné přípony nefungují.

Viz také: **require_once()**, **include_once()**, **get_required_files()**

LIX. POSIX functions

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. POSIX.1 for example defined the open(), read(), write() and close() functions, too, which traditionally have been part of PHP 3 for a long time. Some more system specific functions have not been available before, though, and this module tries to remedy this by providing easy access to these functions.

posix_kill (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Send a signal to a process

```
bool posix_kill (int pid, int sig)
```

Send the signal *sig* to the process with the process identifier *pid*. Returns FALSE, if unable to send the signal, TRUE otherwise.

See also the kill(2) manual page of your POSIX system, which contains additional information about negative process identifiers, the special pid 0, the special pid -1, and the signal number 0.

posix_getpid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the current process identifier

```
int posix_getpid (void )
```

Return the process identifier of the current process.

posix_getppid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the parent process identifier

```
int posix_getppid (void )
```

Return the process identifier of the parent process of the current process.

posix_getuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the real user ID of the current process

```
int posix_getuid (void )
```

Return the numeric real user ID of the current process. See also **posix_getpwuid()** for information on how to convert this into a useable username.

posix_geteuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the effective user ID of the current process

```
int posix_geteuid (void )
```

Return the numeric effective user ID of the current process. See also **posix_getpwuid()** for information on how to convert this into a useable username.

posix_getgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the real group ID of the current process

```
int posix_getgid (void )
```

Return the numeric real group ID of the current process. See also **posix_getgrgid()** for information on how to convert this into a useable group name.

posix_getegid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the effective group ID of the current process

```
int posix_getegid (void )
```

Return the numeric effective group ID of the current process. See also **posix_getgrgid()** for information on how to convert this into a useable group name.

posix_setuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Set the effective UID of the current process

```
bool posix_setuid (int uid)
```

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns TRUE on success, FALSE otherwise. See also **posix_setgid()**.

posix_setgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Set the effective GID of the current process

```
bool posix_setgid (int gid)
```

Set the real group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function. The appropriate order of function calls is **posix_setgid()** first, **posix_setuid()** last.

Returns TRUE on success, FALSE otherwise.

posix_getgroups (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the group set of the current process

```
array posix_getgroups (void )
```

Returns an array of integers containing the numeric group ids of the group set of the current process. See also **posix_getgrgid()** for information on how to convert this into useable group names.

posix_getlogin (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return login name

```
string posix_getlogin (void )
```

Returns the login name of the user owning the current process. See **posix_getpwnam()** for information how to get more information about this user.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return the current process group identifier

```
int posix_getpgrp (void )
```

Return the process group identifier of the current process. See POSIX.1 and the getpgrp(2) manual page on your POSIX system for more information on process groups.

posix_setsid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Make the current process a session leader

```
int posix_setsid (void )
```

Make the current process a session leader. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns the session id.

posix_setpgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

set process group id for job control

```
int posix_setpgid (int pid, int pgid)
```

Let the process *pid* join the process group *pgid*. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns TRUE on success, FALSE otherwise.

posix_getpgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Get process group id for job control

```
int posix_getpgid (int pid)
```

Returns the process group identifier of the process *pid*.

This is not a POSIX function, but is common on BSD and System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_getsid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Get the current sid of the process

```
int posix_getsid (int pid)
```

Return the sid of the process *pid*. If *pid* is 0, the sid of the current process is returned.

This is not a POSIX function, but is common on System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_uname (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Get system name

```
array posix_uname (void )
```

Returns a hash of strings with information about the system. The indices of the hash are

- sysname - operating system name (e.g. Linux)
- nodename - system name (e.g. valiant)
- release - operating system release (e.g. 2.2.10)
- version - operating system version (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - system architecture (e.g. i586)
- domainname - DNS domainname (e.g. php.net)

domainname is a GNU extension and not part of POSIX.1, so this field is only available on GNU systems or when using the GNU libc.

Posix requires that you must not make any assumptions about the format of the values, e.g. you cannot rely on three digit version numbers or anything else returned by this function.

posix_times (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Get process times

```
array posix_times (void )
```

Returns a hash of strings with information about the current process CPU usage. The indices of the hash are

- ticks - the number of clock ticks that have elapsed since reboot.
- utime - user time used by the current process.
- stime - system time used by the current process.
- cutime - user time used by current process and children.
- cstime - system time used by current process and children.

posix_ctermid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Get path name of controlling terminal

```
string posix_ctermid (void )
```

Needs to be written.

posix_ttyname (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Determine terminal device name

```
string posix_ttyname (int fd)
```

Needs to be written.

posix_isatty (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Determine if a file descriptor is an interactive terminal

```
bool posix_isatty (int fd)
```

Needs to be written.

posix_getcwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Pathname of current directory

```
string posix_getcwd (void )
```

Needs to be written ASAP.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Create a fifo special file (a named pipe)

```
bool posix_mkfifo (string pathname, int mode)
```

Needs to be written ASAP.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return info about a group by name

```
array posix_getgrnam (string name)
```

Needs to be written.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return info about a group by group id

```
array posix_getgrgid (int gid)
```

Needs to be written.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return info about a user by username

```
array posix_getpwnam (string username)
```

Returns an associative array containing information about a user referenced by an alphanumeric username, passed in the *username* parameter.

The array elements returned are:

Tabulka 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name. This should be the same as the *username parameter used when calling the function, and hence redundant.*
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID of the user in numeric form.
gid	The group ID of the user. Use the function **posix_getgrgid() to resolve the group name and a list of its members.**
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return info about a user by user id

```
array posix_getpwuid (int uid)
```

Returns an associative array containing information about a user referenced by a numeric user ID, passed in the *uid* parameter.

The array elements returned are:

Tabulka 1. The user information array

Element	Description

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name.
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID, should be the same as the *`uid parameter used when calling the function, and hence redundant.`*
gid	The group ID of the user. Use the function **posix_getgrgid() to resolve the group name and a list of its members.**
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Return info about system ressource limits

```
array posix_getrlimit (void )
```

Needs to be written ASAP.

LX. PostgreSQL functions

Postgres, developed originally in the UC Berkeley Computer Science Department, pioneered many of the object-relational concepts now becoming available in some commercial databases. It provides SQL92/SQL3 language support, transaction integrity, and type extensibility. PostgreSQL is an open source descendant of this original Berkeley code.

PostgreSQL is available without cost. The current version is available at www.PostgreSQL.org (http://www.postgresql.org/).

Since version 6.3 (03/02/1998) PostgreSQL uses unix domain sockets. A table is shown below describing these new connection possibilities. This socket will be found in `/tmp/.s.PGSQL.5432`. This option can be enabled with the '-i' flag to **postmaster** and it's meaning is: "listen on TCP/IP sockets as well as Unix domain sockets".

Tabulka 1. Postmaster and PHP

Postmaster	PHP	Status
postmaster &	pg_connect("dbname=MyDbName");	OK
postmaster -i &	pg_connect("dbname=MyDbName");	OK
postmaster &	pg_connect("host=localhost dbname=MyDbName");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at 'localhost' on port '5432'? in /path/to/file.php3 on line 20.
postmaster -i &	pg_connect("host=localhost dbname=MyDbName");	OK

One can establish a connection with the following value pairs set in the command string: **$conn = pg_Connect("host=myHost port=myPort tty=myTTY options=myOptions dbname=myDB user=myUser password=myPassword ");**

The previous syntax of: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** has been deprecated.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block. A transaction block starts with a **begin** and if the transaction was valid ends with **commit** or **end**. If the transaction fails the transaction should be closed with **rollback** or **abort**.

Příklad 1. Using Large Objects

```
<?php
    $database = pg_Connect ("dbname=jacarta");
    pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
    pg_exec ($database, "commit");
?>
```

pg_close (PHP 3, PHP 4)

Close a PostgreSQL connection

```
bool pg_close (int connection)
```

Returns false if connection is not a valid connection index, true otherwise. Closes down the connection to a PostgreSQL database associated with the given connection index.

> **Poznámka:** This isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

pg_close() will not close persistent links generated by **pg_pconnect()**.

pg_cmdtuples (PHP 3, PHP 4)

Returns number of affected tuples

```
int pg_cmdtuples (int result_id)
```

Pg_cmdtuples() returns the number of tuples (instances) affected by INSERT, UPDATE, and DELETE queries. If no tuple is affected the function will return 0.

Příklad 1. Pg_cmdtuples()

```
<?php
$result = pg_exec ($conn, "INSERT INTO publisher VALUES ('Author')");
$cmdtuples = pg_cmdtuples ($result);
echo $cmdtuples . " <- cmdtuples affected.";
?>
```

pg_connect (PHP 3, PHP 4)

Open a PostgreSQL connection

```
int pg_connect (string conn_string)
```

Returns a connection index on success, or false if the connection could not be made. Opens a connection to a PostgreSQL database. The arguments should be within a quoted string.

Příklad 1. Using pg_connect arguments

```
<?php
$dbconn = pg_Connect ("dbname=mary");
//connect to a database named "mary"
$dbconn2 = pg_Connect ("host=localhost port=5432 dbname=mary");
//connect to a database named "mary" on "localhost" at port "5432"
$dbconn3 = pg_Connect ("host=sheep port=5432 dbname=mary user=lamb password=baaaa");
//connect to a database named "mary" on the host "sheep" with a username and password
?>
```

The arguments available include *host*, *port*, *tty*, *options*, *dbname*, *user*, and *password*.

This function returns a connection index that is needed by other PostgreSQL functions. You can have multiple connections open at once.

The previous syntax of: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** has been deprecated.

See also **pg_pconnect()**.

pg_dbname (PHP 3, PHP 4)

Get the database name

```
string pg_dbname (int connection)
```

Returns the name of the database that the given PostgreSQL connection index is connected to, or false if connection is not a valid connection index.

pg_end_copy (PHP 4 >= 4.0.3)

Sync with PostgreSQL backend

```
bool pg_end_copy ([resource connection])
```

pg_end_copy() syncs PostgreSQL frontend with the backend after doing a copy operation. It must be issued or the backend may get "out of sync" with the frontend. Returns TRUE if successfull, FALSE otherwise.

For further details and an example, see also **pg_put_line()**.

pg_errormessage (PHP 3, PHP 4)

Get the error message string

```
string pg_errormessage (int connection)
```

Returns a string containing the error message, false on failure. Details about the error probably cannot be retrieved using the **pg_errormessage()** function if an error occured on the last database action for which a valid connection exists, this function will return a string containing the error message generated by the backend server.

pg_exec (PHP 3, PHP 4)

Execute a query

```
int pg_exec (int connection, string query)
```

Returns a result index if query could be executed, false on failure or if connection is not a valid connection index. Details about the error can be retrieved using the **pg_ErrorMessage()** function if connection is valid. Sends an SQL statement to the PostgreSQL database specified by the connection index. The connection must be a valid index that was returned by **pg_Connect()**. The return value of this function is an index to be used to access the results from other PostgreSQL functions.

Poznámka: PHP/FI returned 1 if the query was not expected to return data (inserts or updates, for example) and greater than 1 even on selects that did not return anything. No such assumption can be made in PHP.

pg_fetch_array (PHP 3>= 3.0.1, PHP 4)

Fetch a row as an array

```
array pg_fetch_array (int result, int row [, int result_type])
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

Pg_fetch_array() is an extended version of **pg_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

The third optional argument *result_type* in **pg_fetch_array()** is a constant and can take the following values: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

Poznámka: *Result_type* was added in PHP 4.0.

An important thing to note is that using **pg_fetch_array()** is NOT significantly slower than using **pg_fetch_row()**, while it provides a significant added value.

For further details, see also **pg_fetch_row()**

Příklad 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_fetch_object (PHP 3>= 3.0.1, PHP 4)

Fetch a row as an object

```
object pg_fetch_object (int result, int row [, int result_type])
```

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

Pg_fetch_object() is similar to **pg_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The third optional argument *result_type* in **pg_fetch_object()** is a constant and can take the following values: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

Poznámka: *Result_type* was added in PHP 4.0.

Speed-wise, the function is identical to **pg_fetch_array()**, and almost as quick as **pg_fetch_row()** (the difference is insignificant).

See also: **pg_fetch_array()** and **pg_fetch_row()**.

Příklad 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("host=localhost port=5432 dbname=$database");
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <?php echo $database ?></H1> <?php
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>

<PRE><?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo "-------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "-------\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>
```

pg_fetch_row (PHP 3>= 3.0.1, PHP 4)

Get a row as an enumerated array

```
array pg_fetch_row (int result, int row)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

Pg_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

See also: **pg_fetch_array()**, **pg_fetch_object()**, **pg_result()**.

Příklad 1. Postgres fetch row

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$num = pg_numrows($result);

for ($i=0; $i<$num; $i++) {
  $r = pg_fetch_row($result, $i);

  for ($j=0; $j<count($r); $j++) {
    echo "$r[$j] ";
  }

  echo "<BR>";

}

?>
```

pg_fieldisnull (PHP 3, PHP 4)

Test if a field is NULL

```
int pg_fieldisnull (int result_id, int row, mixed field)
```

Test if a field is NULL or not. Returns 0 if the field in the given row is not NULL. Returns 1 if the field in the given row is NULL. Field can be specified as number or fieldname. Row numbering starts at 0.

pg_fieldname (PHP 3, PHP 4)

Returns the name of a field

```
string pg_fieldname (int result_id, int field_number)
```

Pg_fieldname() will return the name of the field occupying the given column number in the given PostgreSQL result identifier. Field numbering starts from 0.

pg_fieldnum (PHP 3, PHP 4)

Returns the field number of the named field

```
int pg_fieldnum (int result_id, string field_name)
```

Pg_fieldnum() will return the number of the column slot that corresponds to the named field in the given PosgreSQL result identifier. Field numbering starts at 0. This function will return -1 on error.

pg_fieldprtlen (PHP 3, PHP 4)

Returns the printed length

```
int pg_fieldprtlen (int result_id, int row_number, string field_name)
```

Pg_fieldprtlen() will return the actual printed length (number of characters) of a specific value in a PostgreSQL result. Row numbering starts at 0. This function will return -1 on an error.

pg_fieldsize (PHP 3, PHP 4)

Returns the internal storage size of the named field

```
int pg_fieldsize (int result_id, int field_number)
```

Pg_fieldsize() will return the internal storage size (in bytes) of the field number in the given PostgreSQL result. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return false on error.

pg_fieldtype (PHP 3, PHP 4)

Returns the type name for the corresponding field number

```
string pg_fieldtype (int result_id, int field_number)
```

Pg_fieldtype() will return a string containing the type name of the given field in the given PostgreSQL result identifier. Field numbering starts at 0.

pg_freeresult (PHP 3, PHP 4)

Free result memory

```
int pg_freeresult (int result_id)
```

Pg_freeresult() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_freeresult()** with the result identifier as an argument and the associated result memory will be freed.

pg_getlastoid (PHP 3, PHP 4)

Returns the last object identifier

```
int pg_getlastoid (int result_id)
```

Pg_getlastoid() can be used to retrieve the `oid` assigned to an inserted tuple if the result identifier is used from the last command sent via **pg_exec()** and was an SQL INSERT. This function will return a positive integer if there was a valid `oid`. It will return -1 if an error occured or the last command sent via **pg_exec()** was not an INSERT.

pg_host (PHP 3, PHP 4)

Returns the host name associated with the connection

```
string pg_host (int connection_id)
```

Pg_host() will return the host name of the given PostgreSQL connection identifier is connected to.

pg_loclose (PHP 3, PHP 4)

Close a large object

```
void pg_loclose (int fd)
```

Pg_loclose() closes an Inversion Large Object. *Fd* is a file descriptor for the large object from **pg_loopen()**.

pg_locreate (PHP 3, PHP 4)

Create a large object

```
int pg_locreate (int conn)
```

Pg_locreate() creates an Inversion Large Object and returns the `oid` of the large object. *conn* specifies a valid database connection. PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above).

pg_loexport (PHP 4 >= 4.0.1)

Export a large object to file

```
bool pg_loexport (int oid, int file [, int connection_id])
```

The *oid* argument specifies the object id of the large object to export and the *filename* argument specifies the pathname of the file. Returns FALSE if an error occurred, TRUE otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loimport (PHP 4 >= 4.0.1)

Import a large object from file

```
int pg_loimport (int file [, int connection_id])
```

The *filename* argument specifies the pathname of the file to be imported as a large object. Returns FALSE if an error occurred, object id of the just created large object otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loopen (PHP 3, PHP 4)

Open a large object

```
int pg_loopen (int conn, int objoid, string mode)
```

Pg_loopen() open an Inversion Large Object and returns file descriptor of the large object. The file descriptor encapsulates information about the connection. Do not close the connection before closing the large object file descriptor. *objoid* specifies a valid large object oid and *mode* can be either "r", "w", or "rw".

pg_loread (PHP 3, PHP 4)

Read a large object

```
string pg_loread (int fd, int len)
```

pg_loread() reads at most *len* bytes from a large object and returns it as a string. *fd* specifies a valid large object file descriptor and *len* specifies the maximum allowable size of the large object segment.

pg_loreadall (PHP 3, PHP 4)

Read a entire large object and send straight to browser

```
void pg_loreadall (int fd)
```

Pg_loreadall() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound.

pg_lounlink (PHP 3, PHP 4)

Delete a large object

```
void pg_lounlink (int conn, int lobjid)
```

Pg_lounlink() deletes a large object with the *lobjid* identifier for that large object.

pg_lowrite (PHP 3, PHP 4)

Write a large object

```
int pg_lowrite (int fd, string buf)
```

Pg_lowrite() writes at most to a large object from a variable *buf* and returns the number of bytes actually written, or false in the case of an error. *fd* is a file descriptor for the large object from **pg_loopen()**.

pg_numfields (PHP 3, PHP 4)

Returns the number of fields

```
int pg_numfields (int result_id)
```

Pg_numfields() will return the number of fields (columns) in a PostgreSQL result. The argument is a valid result identifier returned by **pg_exec()**. This function will return -1 on error.

pg_numrows (PHP 3, PHP 4)

Returns the number of rows

```
int pg_numrows (int result_id)
```

Pg_numrows() will return the number of rows in a PostgreSQL result. The argument is a valid result identifier returned by **pg_exec()**. This function will return -1 on error.

pg_options (PHP 3, PHP 4)

Get the options associated with the connection

```
string pg_options (int connection_id)
```

Pg_options() will return a string containing the options specified on the given PostgreSQL connection identifier.

pg_pconnect (PHP 3, PHP 4)

Open a persistant PostgreSQL connection

```
int pg_pconnect (string conn_string)
```

Returns a connection index on success, or false if the connection could not be made. Opens a connection to a PostgreSQL database. The arguments should be within a quoted string. The arguments available include *host*, *port*, *tty*, *options*, *dbname*, *user*, and *password*.

This function returns a connection index that is needed by other PostgreSQL functions. You can have multiple connections open at once.

The previous syntax of: **$conn = pg_pconnect ("host", "port", "options", "tty", "dbname")** has been deprecated.

See also **pg_connect()**.

pg_port (PHP 3, PHP 4)

Return the port number associated with the connection

```
int pg_port (int connection_id)
```

Pg_port() will return the port number that the given PostgreSQL connection identifier is connected to.

pg_put_line (PHP 4 >= 4.0.3)

Send a NULL-terminated string to PostgreSQL backend

```
bool pg_put_line ([resource connection_id, string data])
```

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL copy-operation. That final NULL-character is added automatically. Returns TRUE if successfull, FALSE otherwise.

> **Poznámka:** Note the application must explicitly send the two characters "\." on a final line to indicate to the backend that it has finished sending its data.

See also **pg_end_copy()**.

Příklad 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_exec($conn, "create table bar (a int4, b char(16), d float8)");
    pg_exec($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_result (PHP 3, PHP 4)

Returns values from a result identifier

```
mixed pg_result (int result_id, int row_number, mixed fieldname)
```

Pg_result() will return values from a result identifier produced by **pg_Exec()**. The *row_number* and *fieldname* sepcify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of integer, boolean and oid types are returned as integer values. All forms of float, and real types are returned as double values. All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Set the client encoding

```
int pg_set_client_encoding ([int connection, string encoding])
```

The function set the client encoding and return 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Poznámka:** This function requires PHP-4.0.2 or higher and PostgreSQL-7.0 or higher.
>
> The function used to be called **pg_setclientencoding()**.

See also **pg_client_encoding()**.

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Get the client encoding

```
string pg_client_encoding ([int connection])
```

The functions returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Poznámka:** This function requires PHP-4.0.2 or higher and PostgreSQL-7.0 or higher.
>
> The function used to be called **pg_clientencoding()**.

See also **pg_set_client_encoding()**.

pg_trace (PHP 4 >= 4.0.1)

Enable tracing a PostgreSQL connection

```
bool pg_trace (string filename [, string mode [, int connection]])
```

Enables tracing of the PostgreSQL frontend/backend communication to a debugging file. To fully understand the results one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing errors in queries sent to the server, you could do for example **grep '^To backend' trace.log** and see what query actually were sent to the PostgreSQL server.

Filename and *mode* are the same as in **fopen()** (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

Returns TRUE if *filename* could be opened for logging, FALSE otherwise.

See also **fopen()** and **pg_untrace()**.

pg_tty (PHP 3, PHP 4)

Return the tty name associated with the connection

```
string pg_tty (int connection_id)
```

Pg_tty() will return the tty name that server side debugging output is sent to on the given PostgreSQL connection identifier.

pg_untrace (PHP 4 >= 4.0.1)

Disable tracing of a PostgreSQL connection

```
bool pg_untrace ([int connection])
```

Stop tracing started by **pg_trace()**. *connection* specifies the connection that was traced and defaults to the last one opened.

Returns always TRUE.

See also **pg_trace()**.

LXI. Funkce Spouštění Programů

escapeshellarg (PHP 4 >= 4.0.3)

escape a string to be used as a shell argument

```
string escapeshellarg (string arg)
```

EscapeShellArg() přidá jednoduché uvozovky na začátek a konec řetězce a ouvozovkuje/escapes všechny výskyty jednoduchých uvozovek, takže tento řetězec můžete přímo předat shell funkci, přičemž tento bude brán jako bezpečný argument. Tato funkce by se měla používat k oescapování jednotlivých argumentů určených pro shell funkce pocházejících z uživatelského vstupu. Shell funkce zahrnujíc **exec()**, **system()** a backtick operator. Standardní použití:

```
system("ls ".EscapeShellArg($dir))
```

Viz také **exec()**, **popen()**, **system()**, a backtick operátor.

escapeshellcmd (PHP 3, PHP 4)

escape shellovské metaznaky

```
string escapeshellcmd (string command)
```

EscapeShellCmd() oescapuje všechny znaky v řetězci, které by se daly použít ke zneužití shellového příkazu k vykonání libovolných příkazů. Tato funkce by se měla používat k zabezpečení toho, aby všechna data pocházející z uživatelského vstupu byla oescapována přetím, než budou předána funkci **exec()** nebo **system()**, nebo backtick operátoru. Standardní použití:

```
$e = EscapeShellCmd($userinput);
system("echo $e"); // tady nás nezajímá, jestli jsou v $e mezery
$f = EscapeShellCmd($filename);
system("touch \"/tmp/$f\"; ls -l \"/tmp/$f\""); // a tady ano, proto použijeme uvozovky
```

Viz také**escapeshellarg()**, **exec()**, **popen()**, **system()**, a backtick operátor.

exec (PHP 3, PHP 4)

Provést externí program

```
string exec (string command [, string array [, int return_var]])
```

exec() provádí předaný *command*, nicméně nic netiskne. Pouze vrací poslední řádek výstupu daného příkazu. Pokud potřebujete provést příkaz a nechat všechna data z tohoto příkazu předat rovnou bez jakéhokoli zásahu, použijte funkci **PassThru()**.

Pokud je přítomen argument *array*, předané pole se naplní všemi řádky výstupu daného příkazu. Pozn.: Pokud toto pole už obsahuje nějaké prvky, **exec()** připojí tento výstup na konec tohoto pole. Pokud nechcete, aby tato funkce připojovala prvky na konec daného pole, zavolejte na toto pole **unset()** předtím, než ho předáte funkci **exec()**.

Pokud je vedle argumentu *array* přítomen argument *return_var*, návratová hodnota provedeného příkazu se zapíše do této proměnné.

Pozn.: Pokud chcete používat v této funkci data z uživatelského vstupu, měli byste používat **EscapeShellCmd()**, abyste měli jistotu, že uživatelé nevmanipulují systém do provádění libovolných příkazů.

Pozn.: Pokud touto funkcí nastartujete nějaký program a chcete ho nechat běžet v pozadí, musíte se zajistit přesměrování výstupu z tohoto programu do souboru nebo jineho výstupního streamu, jinak se PHP zasekne až do ukončení běhu tohoto programu.

Viz také**system()**, **PassThru()**, **popen()**, **EscapeShellCmd()**, a backtick operátor.

passthru (PHP 3, PHP 4)

Vykonat externí program a zobrazit nezpracovaný výstup

```
void passthru (string command [, int return_var])
```

Funkce **passthru()** se podobá funkci **Exec()** v tom ohledu, že provede *command*. Pokud je přítomen argument *return_var*, návratová hodnota tohoto příkazu se umístí sem. Tato funkce by se měla používat místo **Exec()** a **System()**, pokud jsou výstupem daného příkazu binární data, která je potřeba odeslat přímo do browseru. Běžným použitím této funkce vykonat např. pbmplus utility, které mohou poslat stream obrázku na stdout. Nastavením content-type na *image/gif* a zavoláním pbmplus programu k odeslání gifu na stdout gifu můžete vytvořit PHP skripty, které přímo tvoří obrázky.

Pozn.: Pokud touto funkcí nastartujete nějaký program a chcete ho nechat běžet v pozadí, musíte se zajistit přesměrování výstupu z tohoto programu do souboru nebo jineho výstupního streamu, jinak se PHP zasekne až do ukončení běhu tohoto programu.

Viz také**exec()**, **system()**, **popen()**, **EscapeShellCmd()**, a backtick operátor.

system (PHP 3, PHP 4)

Provést externí program a zobrazit výstup

```
string system (string command [, int return_var])
```

system() je verzí stejnojmenné C funkce; vykoná předaný *command* a zobrazí výstup. Pokud jí předáte proměnnou jako druhý argument, návratová hodnota provedeného příkazu se zapíše do této proměnné.

Pozn.: Pokud chcete používat v této funkci data z uživatelského vstupu, měli byste používat **EscapeShellCmd()**, abyste měli jistotu, že uživatelé nevmanipulují systém do provádění libovolných příkazů.

Pozn.: Pokud touto funkcí nastartujete nějaký program a chcete ho nechat běžet v pozadí, musíte se zajistit přesměrování výstupu z tohoto programu do souboru nebo jineho výstupního streamu, jinak se PHP zasekne až do ukončení běhu tohoto programu.

Pokud PHP běží jako modul serveru, **system()** také automaticky flushne výstupní buffer web serveru po každém řádku výstupu.

Při úspěchu vrací poslední řádek výstupu příkazu, při selhání `false`.

Pokud potřebujete provést příkaz a nechat všechna data z tohoto příkazu předat rovnou bez jakéhokoli zásahu, použijte funkci **PassThru()**.

Viz také**exec()**, **PassThru()**, **popen()**, **EscapeShellCmd()**, a backtick operátor.

LXII. Pspell Functions

These functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://pspell.sourceforge.net/ respectively, and add the `-with-pspell[=dir]` option when compiling php.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Add the word to a personal wordlist.

```
int pspell_add_to_personal (int dictionary_link, string word)
```

Pspell_add_to_personal() adds a word to the personal wordlist. If you used **pspell_new_config()** with **pspell_config_personal()** to open the dictionary, you can save the wordlist later with **pspell_save_wordlist()**. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Příklad 1. Pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Add the word to the wordlist in the current session.

```
int pspell_add_to_session (int dictionary_link, string word)
```

Pspell_add_to_session() adds a word to the wordlist associated with the current session. It is very similar to **pspell_add_to_personal()**

pspell_check (PHP 4 >= 4.0.2)

Check a word

```
boolean pspell_check (int dictionary_link, string word)
```

Pspell_check() checks the spelling of a word and returns true if the spelling is correct, false if not.

Příklad 1. Pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session (PHP 4 >= 4.0.2)

Clear the current session.

```
int pspell_clear_session (int dictionary_link)
```

Pspell_clear_session() clears the current session. The current wordlist becomes blank, and, for example, if you try to save it with **pspell_save_wordlist()**, nothing happens.

Příklad 1. Pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link); //"Vlad" will not be saved
```

pspell_config_create (PHP 4 >= 4.0.2)

Create a config used to open a dictionary.

```
int pspell_config_create (string language [, string spelling [, string jargon [, string
encoding]]])
```

Pspell_config_create() has a very similar syntax to **pspell_new()**. In fact, using **pspell_config_create()** immediatelly followed by **pspell_new_config()** will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling **pspell_new_config()** to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- PSPELL_FAST - Fast mode (least number of suggestions)
- PSPELL_NORMAL - Normal mode (more suggestions)
- PSPELL_BAD_SPELLERS - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Příklad 1. Pspell_config_create()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore words less than N characters long.

```
int pspell_config_ignore (int dictionary_link, int n)
```

Pspell_config_ignore() should be used on a config before calling **pspell_new_config()**. This function allows short words to be skipped by the spellchecker. Words less then n characters will be skipped.

Příklad 1. Pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd"); //will not result in an error
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change the mode number of suggestions returned.

```
int pspell_config_mode (int dictionary_link, int mode)
```

Pspell_config_mode() should be used on a config before calling **pspell_new_config()**. This function determines how many suggestions will be returned by **pspell_suggest()**.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- PSPELL_FAST - Fast mode (least number of suggestions)
- PSPELL_NORMAL - Normal mode (more suggestions)
- PSPELL_BAD_SPELLERS - Slow mode (a lot of suggestions)

Příklad 1. Pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal (PHP 4 >= 4.0.2)

Set a file that contains personal wordlist.

```
int pspell_config_personal (int dictionary_link, string file)
```

Pspell_config_personal() should be used on a config before calling **pspell_new_config()**. The personal wordlist will be loaded and used in addition to the standard one after you call **pspell_new_config()**. If the file does not exist, it will be created. The file is also the file where **pspell_save_wordlist()** will save personal wordlist to. The file should be

writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Příklad 1. Pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl (PHP 4 >= 4.0.2)

Set a file that contains replacement pairs.

```
int pspell_config_repl (int dictionary_link, string file)
```

Pspell_config_repl() should be used on a config before calling **pspell_new_config()**. The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, **pspell_store_replacement()** can be used to store a replacement pair and then **pspell_save_wordlist()** to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Příklad 1. Pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Consider run-together words as valid compounds.

```
int pspell_config_runtogether (int dictionary_link, boolean flag)
```

Pspell_config_runtogether() should be used on a config before calling **pspell_new_config()**. This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Příklad 1. Pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Determine whether to save a replacement pairs list along with the wordlist.

```
int pspell_config_save_repl (int dictionary_link, boolean flag)
```

Pspell_config_save_repl() should be used on a config before calling **pspell_new_config()**. It determines whether **pspell_save_wordlist()** will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if **pspell_config_repl()** is used, the replacement pairs will be saved by **pspell_save_wordlist()** anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

```
int pspell_new (string language [, string spelling [, string jargon [, string encoding
[, int mode]]]])
```

Pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- PSPELL_FAST - Fast mode (least number of suggestions)
- PSPELL_NORMAL - Normal mode (more suggestions)
- PSPELL_BAD_SPELLERS - Slow mode (a lot of suggestions)
- PSPELL_RUN_TOGETHER - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, PSPELL_FAST, PSPELL_NORMAL and PSPELL_BAD_SPELLERS are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Příklad 1. Pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new_config (PHP 4 >= 4.0.2)

Load a new dictionary with settings based on a given config

```
int pspell_new_config (int config)
```

Pspell_new_config() opens up a new dictionary with settings specified in a config, created with with **pspell_config_create()** and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by **pspell_new()** and **pspell_new_personal()**.

The config parameter is the one returned by **pspell_config_create()** when the config was created.

Příklad 1. Pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_new_personal (PHP 4 >= 4.0.2)

Load a new dictionary with personal wordlist

```
int pspell_new_personal (string personal, string language [, string spelling [, string
jargon [, string encoding [, int mode]]]])
```

Pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with **pspell_save_wordlist()**, if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using **pspell_config_create()**, set the personal wordlist file with **pspell_config_personal ()**, set the file for replacement pairs with **pspell_config_repl()**, and open a new dictionary with **pspell_new_config()**.

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- PSPELL_FAST - Fast mode (least number of suggestions)
- PSPELL_NORMAL - Normal mode (more suggestions)
- PSPELL_BAD_SPELLERS - Slow mode (a lot of suggestions)
- PSPELL_RUN_TOGETHER - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, PSPELL_FAST, PSPELL_NORMAL and PSPELL_BAD_SPELLERS are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Příklad 1. Pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
"en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Save the personal wordlist to a file.

```
int pspell_save_wordlist (int dictionary_link)
```

Pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with **pspell_new_personal()**, and the location of files to be saved specified with **pspell_config_personal()** and (optionally) **pspell_config_repl()**. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Příklad 1. Pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Store a replacement pair for a word

```
int pspell_store_replacement (int dictionary_link, string misspelled, string correct)
```

Pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by **pspell_suggest()** later. In order to be able to take advantage of this function, you have to use **pspell_new_personal()** to open the dictionary. In order to permanently save the replacement pair, you have to use **pspell_config_personal()** and **pspell_config_repl()** to set the path where to save your custom wordlists, and then use **pspell_save_wordlist()** for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Příklad 1. Pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

```
array pspell_suggest (int dictionary_link, string word)
```

Pspell_suggest() returns an array of possible spellings for the given word.

Příklad 1. Pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Possible spelling: " . $suggestions[$i] . "<br>";
    }
}
```

LXIII. GNU Readline

readline() funkce implementují rozhraní ke GNU Readline knihovně. Tyto funkce poskytují editovatelné příkazové řádky. Příkladem může být způsob, jakým vám Bash umožňuje vkládat znaky nebo listovat historií příkazů pomocí kláves pro pohyb kurzoru. Z interaktivní povahy této knihovny vyplývá, že není příliš užitečná pro psaní webových aplikací, ale může být užitečná při psaní skriptů, které se mají spouštět z příkazové řádky.

Domovskou stránkou GNU Readline projektu je http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Udržuje jej Chet Ramey, který je také autorem Bashe.

readline (PHP 4 >= 4.0b4)

Přečíst řádek

```
string readline ([string prompt])
```

Tato funkce vrátí jeden řádek od uživatele. Můžete specifikovat řetězec, který se má uživateli zobrazit jako výzva. Z vráceného řádku je odstraněna sekvence konce řádku. Do historie tento řádek musíte přidat sami pomocí **readline_add_history()**.

Příklad 1. Readline()

```
//ziska 3 prikazy uzivatele
for ($i=0; $i < 3; $i++) {
        $line = readline ("Command: ");
        readline_add_history ($line);
}

//vypise historii
print_r (readline_list_history());

//vypise promenne
print_r (readline_info());
```

readline_add_history (PHP 4 >= 4.0b4)

Přidat řádek do historie

```
void readline_add_history (string line)
```

Tato funkce přidá řádek do historie příkazové řádky.

readline_clear_history (PHP 4 >= 4.0b4)

Smazat historii

```
boolean readline_clear_history (void )
```

Tato funkce smaže celou historii příkazové řádky.

readline_completion_function (PHP 4 >= 4.0b4)

Zaregistrovat dokončující funkci

```
boolean readline_completion_function  (string line)
```

Tato funkce zaregistruje dokončující funkci. Musíte zadat název existující funkce, která přijímá částečný příkaz a vrací pole možných protějšků. Toto je stejná funkcionalita jakou dostanete, pokud v Bashi zmáčknete klávesu tab.

readline_info (PHP 4 >= 4.0b4)

Zjistit/nastavit různé interní proměnné readline

```
mixed readline_info ([string varname [, string newvalue]])
```

Při volání bez argumentů tato funkce vrátí pole hodnot pro všechna nastavení, která readline používá. Prvky jsou indexovány podle následujících hodnot: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name a terminal_name.

Při volání s jedním argumentem vrátí hodnotu tohoto nastavení. Při volání se dvěma argumenty se nastavení zmení na danou hodnotu.

readline_list_history (PHP 4 >= 4.0b4)

Vypsat historii

```
array readline_list_history (void )
```

Tato funkce vrátí pole celé historie příkazové řádky. Prvky jsou indexovány integery, počínaje nulou.

readline_read_history (PHP 4 >= 4.0b4)

Vrátit historii

```
boolean readline_read_history  (string filename)
```

Tato funkce přečte historii příkazové řádky ze souboru.

readline_write_history (PHP 4 >= 4.0b4)

Zapsat historii

```
boolean readline_write_history  (string filename)
```

Tato funkce zapíše historii příkazové řádky do souboru.

LXIV. GNU Recode funkce

Tento modul obsahuje rozhraní ke GNU Recode knihovně, verze 3.5. Pokud cchete používat funkce definované v tomto modulu, musíte zkompilovat váš PHP interpretr s –with-recode. K tomu potřebujete mít na svém systému nainstalovanou GNU Recode 3.5 nebo vyšší.

GNU Recode knihovna konvertuje soubory mezi různými znakovými sadami a kódováními. Když nelze dosáhnout přesné konverze, může se uchýlit k approximacím. Tato knihovna rozeznává nebo produkuje téměř 150 různých znakových sad, a je schopná konvertovat soubory mezi téměř libovolným párem. Podporuje většinu znakových sad z RFC 1345.

recode_string (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Překóduje řetězec podle požadavku

```
string recode_string (string request, string string)
```

Překóduje řetězec *string* podle požadavku *request*. Vrací překódovaný řetězec nebo FALSE, pokud nebylo možné provést požadavek na překódování.

Jednoduchý požadavek na překódování může být "lat1..iso646-de". Detailní instrukce k požadavkům viz GNU Recode dokumentaci u vaší instalace.

Příklad 1. Základní recode_string() příklad:

```
print recode_string ("us..flat", "Následující znak má diakritické znaménko: &aacute;");
```

recode (PHP 4 >= 4.0RC1)

Překóduje řetězec podle požadavku

```
string recode_string (string request, string string)
```

Poznámka: Toto je alias k **recode_string()**. Byl přidán v PHP 4.

recode_file (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Překóduje soubor na soubor podle požadavku

```
boolean recode_file (string request, resource input, resource output)
```

Překóduje soubor odkazovaný deskriptorem *input* do souboru odkazovaném deskriptorem *output* podle požadavku *request*. Pokud se nepodařilo požadavek provést, vrací FALSE, jinak TRUE.

Tato funkce v současnosti nezpracovává deskriptory odkazující na vzdálené soubory (URL). Oba deskriptory musí ukazovat na místní soubory.

Příklad 1. Základní příklad recode_file()

```
$input = fopen (’input.txt’, ’r’);
$output = fopen (’output.txt’, ’w’);
recode_file ("us..flat", $input, $output);
```

LXV. Regular Expression Functions (Perl-Compatible)

The syntax for patterns used in these functions closely resembles Perl. The expression should be enclosed in the delimiters, a forward slash (/), for example. Any character can be used for delimiter as long as it's not alphanumeric or backslash (\). If the delimiter character has to be used in the expression itself, it needs to be escaped by backslash. Since PHP 4.0.4, you can also use Perl-style (), {}, [], and <> matching delimiters.

The ending delimiter may be followed by various modifiers that affect the matching. See Pattern Modifiers.

Příklad 1. Examples of valid patterns

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/
- {^\s+(\s+)?$}

Příklad 2. Examples of invalid patterns

- /href='(.*)' - missing ending delimiter
- /\w+\s*\w+/J - unknown modifier 'J'
- 1-\d3-\d3-\d4| - missing starting delimiter

Poznámka: The Perl-compatible regular expression functions are available in PHP 4 and in PHP 3.0.9 and up.

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. It is available at ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

preg_match (PHP 3>= 3.0.9, PHP 4)

Perform a regular expression match

```
int preg_match (string pattern, string subject [, array matches])
```

Searches *subject* for a match to the regular expression given in *pattern*.

If *matches* is provided, then it is filled with the results of search. $matches[0] will contain the text that match the full pattern, $matches[1] will have the text that matched the first captured parenthesized subpattern, and so on.

Returns true if a match for *pattern* was found in the subject string, or false if not match was found or an error occurred.

Příklad 1. Find the string of text "php"

```
// the "i" after the pattern delimiter indicates a case-insensitive search
if (preg_match ("/php/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Příklad 2. find the word "web"

```
// the \b in the pattern indicates a word boundary, so only the distinct
// word "web" is matched, and not a word partial like "webbing" or "cobweb"
if (preg_match ("/\bweb\b/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
if (preg_match ("/\bweb\b/i", "PHP is the website scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Příklad 3. Getting the domain name out of a URL

```
// get host name from URL
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$host = $matches[2];
// get last two segments of host name
preg_match("/[^\.\/]+\.[^\.\/]+$/",$host,$matches);
echo "domain name is: ".$matches[0]."\n";
```

This example will produce:

```
domain name is: php.net
```

See also **preg_match_all()**, **preg_replace()**, and **preg_split()**.

preg_match_all (PHP 3>= 3.0.9, PHP 4)

Perform a global regular expression match

```
int preg_match_all (string pattern, string subject, array matches [, int order])
```

Searches *subject* for all matches to the regular expression given in *pattern* and puts them in *matches* in the order specified by *order*.

After the first match is found, the subsequent searches are continued on from end of the last match.

order can be one of two things:

PREG_PATTERN_ORDER

Orders results so that $matches[0] is an array of full pattern matches, $matches[1] is an array of strings matched by the first parenthesized subpattern, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_PATTERN_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

This example will produce:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

So, $out[0] contains array of strings that matched full pattern, and $out[1] contains array of strings enclosed by tags.

PREG_SET_ORDER

Orders results so that $matches[0] is an array of first set of matches, $matches[1] is an array of second set of matches, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_SET_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

This example will produce:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

In this case, $matches[0] is the first set of matches, and $matches[0][0] has text matched by full pattern, $matches[0][1] has text matched by first subpattern and so on. Similarly, $matches[1] is the second set of matches, etc.

If *order* is not specified, it is assumed to be PREG_PATTERN_ORDER.

Returns the number of full pattern matches, or false if no match is found or an error occurred.

Příklad 1. Getting all phone numbers out of some text.

```
preg_match_all ("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
                "Call 555-1212 or 1-800-555-1212", $phones);
```

Příklad 2. Find matching HTML tags (greedy)

```
// The \\2 is an example of backreferencing. This tells pcre that
// it must match the second set of parentheses in the regular expression
// itself, which would be the ([\w]+) in this case. The extra backslash is
```

```
// required because the string is in double quotes.
$html = "<b>bold text</b><a href=howdy.html>click me</a>

preg_match_all ("/(<([\w]+)[^>]*>)(.*)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {
echo "matched: ".$matches[0][$i]."\n";
echo "part 1: ".$matches[1][$i]."\n";
echo "part 2: ".$matches[3][$i]."\n";
echo "part 3: ".$matches[4][$i]."\n\n";
}
```

This example will produce:

```
matched: <b>bold text</b>
part 1: <b>
part 2: bold text
part 3: </b>

matched: <a href=howdy.html>click me</a>
part 1: <a href=howdy.html>
part 2: click me
part 3: </a>
```

See also **preg_match()**, **preg_replace()**, and **preg_split()**.

preg_replace (PHP 3>= 3.0.9, PHP 4)

Perform a regular expression search and replace

```
mixed preg_replace (mixed pattern, mixed replacement, mixed subject [, int limit])
```

Searches *subject* for matches to *pattern* and replaces them with *replacement*. If *limit* is specified, then only *limit* matches will be replaced; if *limit* is omitted or is -1, then all matches are replaced.

Replacement may contain references of the form *n* or (since PHP 4.0.4) *$n*, with the latter form being the preferred one. Every such reference will be replaced by the text captured by the *n*'th parenthesized pattern. *n* can be from 0 to 99, and \\0 or $0 refers to the text matched by the whole pattern. Opening parentheses are counted from left to right (starting from 1) to obtain the number of the capturing subpattern.

If matches are found, the new *subject* will be returned, otherwise *subject* will be returned unchanged.

Every parameter to **preg_replace()** can be an array.

If *subject* is an array, then the search and replace is performed on every entry of *subject*, and the return value is an array as well.

If *pattern* and *replacement* are arrays, then **preg_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replacement* has fewer values than *pattern*, then empty string is used for the rest of replacement values. If *pattern* is an array and *replacement* is a string; then this replacement string is used for every value of *pattern*. The converse would not make sense, though.

/e modifier makes **preg_replace()** treat the *replacement* parameter as PHP code after the appropriate references substitution is done. Tip: make sure that *replacement* constitutes a valid PHP code string, otherwise PHP will complain about a parse error at the line containing **preg_replace()**.

Příklad 1. Replacing several values

```
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
```

```
print preg_replace ($patterns, $replace, "{startDate} = 1999-5-27");
```

This example will produce:

```
$startDate = 5/27/1999
```

Příklad 2. Using /e modifier

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
              "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

This would capitalize all HTML tags in the input text.

Příklad 3. Convert HTML to text

```
// $document should contain an HTML document.
// This will remove HTML tags, javascript sections
// and white space. It will also convert some
// common HTML entities to their text equivalent.

$search = array ("'<script[^>]*?>.*?</script>'si",  // Strip out javascript
                 "'<[\/\!]*?[^<>]*?>'si",  // Strip out html tags
                 "'([\r\n])[\s]+'",  // Strip out white space
                 "'&(quot|#34);'i",  // Replace html entities
                 "'&(amp|#38);'i",
                 "'&(lt|#60);'i",
                 "'&(gt|#62);'i",
                 "'&(nbsp|#160);'i",
                 "'&(iexcl|#161);'i",
                 "'&(cent|#162);'i",
                 "'&(pound|#163);'i",
                 "'&(copy|#169);'i",
                 "'&#(\d+);'e");  // evaluate as php

$replace = array ("",
                  "",
                  "\\1",
                  "\"",
                  "&",
                  "<",
                  ">",
                  " ",
                  chr(161),
                  chr(162),
                  chr(163),
                  chr(169),
                  "chr(\\1)");

$text = preg_replace ($search, $replace, $document);
```

Poznámka: Parameter *`limit`* was added after PHP 4.0.1pl2.

See also **preg_match()**, **preg_match_all()**, and **preg_split()**.

preg_replace_callback (PHP 4 CVS only)

Perform a regular expression search and replace using a callback

```
mixed preg_replace_callback (mixed pattern, mixed callback, mixed subject [, int limit])
```

The behavior of this function is almost identical to **preg_replace()**, except for the fact that instead of *replacement* parameter, once should specify a *callback* that will be called and passed an array of matched elements in the subject string. The callback should return the replacement string. This function was added in PHP 4.0.5.

See also **preg_replace()**.

preg_split (PHP 3>= 3.0.9, PHP 4)

Split string by a regular expression

```
array preg_split (string pattern, string subject [, int limit [, int flags]])
```

Poznámka: Parameter *flags* was added in PHP 4 Beta 3.

Returns an array containing substrings of *subject* split along boundaries matched by *pattern*.

If *limit* is specified, then only substrings up to *limit* are returned, and if *limit* is -1, it actually means "no limit", which is useful for specifying the *flags*.

flags can be any combination of the following flags (combined with bitwise | operator):

PREG_SPLIT_NO_EMPTY

If this flag is set, only non-empty pieces will be returned by **preg_split()**.

PREG_SPLIT_DELIM_CAPTURE

If this flag is set, parenthesized expression in the delimiter pattern will be captured and returned as well. This flag was added for 4.0.5.

Příklad 1. preg_split() example

Get the parts of a search string.

```
// split the phrase by any number of commas or space characters,
// which include " ", \r, \t, \n and \f
$keywords = preg_split ("/[\s,]+/", "hypertext language, programming");
```

Splitting a string into component characters.

```
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
```

See also **preg_match()**, **preg_match_all()**, and **preg_replace()**.

preg_quote (PHP 3>= 3.0.9, PHP 4)

Quote regular expression characters

```
string preg_quote (string str [, string delimiter])
```

preg_quote() takes *str* and puts a backslash in front of every character that is part of the regular expression syntax. This is useful if you have a run-time string that you need to match in some text and the string may contain special regex characters.

If the optional *delimiter* is specified, it will also be escaped. This is useful for escaping the delimiter that is required by the PCRE functions. The / is the most commonly used delimiter.

The special regular expression characters are:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Příklad 1.

```
$keywords = "$40 for a g3/400";
$keywords = preg_quote ($keywords, "/");
echo $keywords; // returns \$40 for a g3\/400
```

Příklad 2. Italicizing a word within some text

```
// In this example, preg_quote($word) is used to keep the
// asterisks from having special meaning to the regular
// expression.

$textbody = "This book is *very* difficult to find.";
$word = "*very*";
$textbody = preg_replace ("/".preg_quote($word)."/",
                          "<i>".$word."</i>",
                          $textbody);
```

preg_grep (PHP 4)

Return array entries that match the pattern

```
array preg_grep (string pattern, array input)
```

preg_grep() returns the array consisting of the elements of the *input* array that match the given *pattern*.

Since PHP 4.0.4, the results returned by **preg_grep()** are indexed using the keys from the input array. If this behavior is undesirable, use **array_values()** on the array returned by **preg_grep()** to reindex the values.

Příklad 1. preg_grep() example

```
// return all array elements
// containing floating point numbers
$fl_array = preg_grep ("/^(\d+)?\.\d+$/", $array);
```

Pattern Modifiers (unknown)

Describes possible modifiers in regex patterns

The current possible PCRE modifiers are listed below. The names in parentheses refer to internal PCRE names for these modifiers.

i (PCRE_CASELESS)

If this modifier is set, letters in the pattern match both upper and lower case letters.

m (PCRE_MULTILINE)

By default, PCRE treats the subject string as consisting of a single "line" of characters (even if it actually contains several newlines). The "start of line" metacharacter (^) matches only at the start of the string, while the "end of line" metacharacter ($) matches only at the end of the string, or before a terminating newline (unless *E* modifier is set). This is the same as Perl.

When this modifier is set, the "start of line" and "end of line" constructs match immediately following or immediately before any newline in the subject string, respectively, as well as at the very start and end. This is equivalent to Perl's /m modifier. If there are no "\n" characters in a subject string, or no occurrences of ^ or $ in a pattern, setting this modifier has no effect.

s (PCRE_DOTALL)

If this modifier is set, a dot metacharater in the pattern matches all characters, including newlines. Without it, newlines are excluded. This modifier is equivalent to Perl's /s modifier. A negative class such as [^a] always matches a newline character, independent of the setting of this modifier.

x (PCRE_EXTENDED)

If this modifier is set, whitespace data characters in the pattern are totally ignored except when escaped or inside a character class, and characters between an unescaped # outside a character class and the next newline character, inclusive, are also ignored. This is equivalent to Perl's /x modifier, and makes it possible to include comments inside complicated patterns. Note, however, that this applies only to data characters. Whitespace characters may never appear within special character sequences in a pattern, for example within the sequence (?(which introduces a conditional subpattern.

e

If this modifier is set, **preg_replace()** does normal substitution of backreferences in the replacement string, evaluates it as PHP code, and uses the result for replacing the search string.

Only **preg_replace()** uses this modifier; it is ignored by other PCRE functions.

A (PCRE_ANCHORED)

If this modifier is set, the pattern is forced to be "anchored", that is, it is constrained to match only at the start of the string which is being searched (the "subject string"). This effect can also be achieved by appropriate constructs in the pattern itself, which is the only way to do it in Perl.

D (PCRE_DOLLAR_ENDONLY)

If this modifier is set, a dollar metacharacter in the pattern matches only at the end of the subject string. Without this modifier, a dollar also matches immediately before the final character if it is a newline (but not before any other newlines). This modifier is ignored if *m* modifier is set. There is no equivalent to this modifier in Perl.

S

When a pattern is going to be used several times, it is worth spending more time analyzing it in order to speed up the time taken for matching. If this modifier is set, then this extra analysis is performed. At present, studying a pattern is useful only for non-anchored patterns that do not have a single fixed starting character.

U (PCRE_UNGREEDY)

This modifier inverts the "greediness" of the quantifiers so that they are not greedy by default, but become greedy if followed by "?". It is not compatible with Perl. It can also be set by a (?U) modifier setting within the pattern.

X (PCRE_EXTRA)

This modifier turns on additional functionality of PCRE that is incompatible with Perl. Any backslash in a pattern that is followed by a letter that has no special meaning causes an error, thus reserving these combinations for future expansion. By default, as in Perl, a backslash followed by a letter with no special meaning is treated as a literal. There are at present no other features controlled by this modifier.

Pattern Syntax (unknown)

Describes PCRE regex syntax

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5, with just a few differences (see below). The current implementation corresponds to Perl 5.005.

The differences described here are with respect to Perl 5.005.

1. By default, a whitespace character is any character that the C library function isspace() recognizes, though it is possible to compile PCRE with alternative character type tables. Normally isspace() matches space, formfeed, newline, carriage return, horizontal tab, and vertical tab. Perl 5 no longer includes vertical tab in its set of whitespace characters. The \v escape that was in the Perl documentation for a long time was never in fact recognized. However, the character itself was treated as whitespace at least up to 5.002. In 5.004 and 5.005 it does not match \s.

2. PCRE does not allow repeat quantifiers on lookahead assertions. Perl permits them, but they do not mean what you might think. For example, (?!a){3} does not assert that the next three characters are not "a". It just asserts that the next character is not "a" three times.

3. Capturing subpatterns that occur inside negative lookahead assertions are counted, but their entries in the offsets vector are never set. Perl sets its numerical variables from any such patterns that are matched before the assertion fails to match something (thereby succeeding), but only if the negative lookahead assertion contains just one branch.

4. Though binary zero characters are supported in the subject string, they are not allowed in a pattern string because it is passed as a normal C string, terminated by zero. The escape sequence "\0" can be used in the pattern to represent a binary zero.

5. The following Perl escape sequences are not supported: \l, \u, \L, \U, \E, \Q. In fact these are implemented by Perl's general string-handling and are not part of its pattern matching engine.

6. The Perl \G assertion is not supported as it is not relevant to single pattern matches.

7. Fairly obviously, PCRE does not support the (?{code}) construction.

8. There are at the time of writing some oddities in Perl 5.005_02 concerned with the settings of captured strings when part of a pattern is repeated. For example, matching "aba" against the pattern /^(a(b)?)+$/ sets $2 to the value

"b", but matching "aabbaa" against /^(aa(bb)?)+$/ leaves $2 unset. However, if the pattern is changed to /^(aa(b(b))?)+$/ then $2 (and $3) get set.

In Perl 5.004 $2 is set in both cases, and that is also true of PCRE. If in the future Perl changes to a consistent state that is different, PCRE may change to follow.

9. Another as yet unresolved discrepancy is that in Perl 5.005_02 the pattern /^(a)?(?(1)a|b)+$/ matches the string "a", whereas in PCRE it does not. However, in both Perl and PCRE /^(a)?a/ matched against "a" leaves $1 unset.

10. PCRE provides some extensions to the Perl regular expression facilities:

(a) Although lookbehind assertions must match fixed length strings, each alternative branch of a lookbehind assertion can match a different length of string. Perl 5.005 requires them all to have the same length.

(b) If PCRE_DOLLAR_ENDONLY is set and PCRE_MULTILINE is not set, the $ meta- character matches only at the very end of the string.

(c) If PCRE_EXTRA is set, a backslash followed by a letter with no special meaning is faulted.

(d) If PCRE_UNGREEDY is set, the greediness of the repetition quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

Introduction

The syntax and semantics of the regular expressions supported by PCRE are described below. Regular expressions are also described in the Perl documentation and in a number of other books, some of which have copious examples. Jeffrey Friedl’s "Mastering Regular Expressions", published by O’Reilly (ISBN 1-56592-257-3), covers them in great detail. The description here is intended as reference documentation.

A regular expression is a pattern that is matched against a subject string from left to right. Most characters stand for themselves in a pattern, and match the corresponding characters in the subject. As a trivial example, the pattern

```
The quick brown fox
```

matches a portion of a subject string that is identical to itself.

Meta-caracters

The power of regular expressions comes from the ability to include alternatives and repetitions in the pattern. These are encoded in the pattern by the use of *meta-*

characters, which do not stand for themselves but instead are interpreted in some special way.

There are two different sets of meta-characters: those that are recognized anywhere in the pattern except within square brackets, and those that are recognized in square brackets. Outside square brackets, the meta-characters are as follows:

```
  \      general escape character with several uses
  ^      assert start of subject (or line, in multiline
mode)
  $      assert end of subject (or line, in multiline mode)
  .      match any character except newline (by default)
  [      start character class definition
  |      start of alternative branch
  (      start subpattern
  )      end subpattern
  ?      extends the meaning of (
         also 0 or 1 quantifier
         also quantifier minimizer
  *      0 or more quantifier
  +      1 or more quantifier
  {      start min/max quantifier
```

Part of a pattern that is in square brackets is called a "character class". In a character class the only meta-characters are:

```
  \      general escape character
  ^      negate the class, but only if the first character
  -      indicates character range
  ]      terminates the character class
```

The following sections describe the use of each of the meta-characters.

backslash

The backslash character has several uses. Firstly, if it is followed by a non-alphameric character, it takes away any special meaning that character may have. This use of backslash as an escape character applies both inside and outside character classes.

For example, if you want to match a "*" character, you write "*" in the pattern. This applies whether or not the following character would otherwise be interpreted as a meta-character, so it is always safe to precede a non-alphameric with "\" to specify that it stands for itself. In particular, if you want to match a backslash, you write "\\".

If a pattern is compiled with the PCRE_EXTENDED option, whitespace in the pattern (other than in a character class) and characters between a "#" outside a character class and the next newline character are ignored. An escaping backslash can be used to include a whitespace or "#" character as part of the pattern.

A second use of backslash provides a way of encoding non-

printing characters in patterns in a visible manner. There is no restriction on the appearance of non-printing characters, apart from the binary zero that terminates a pattern, but when a pattern is being prepared by text editing, it is usually easier to use one of the following escape sequences than the binary character it represents:

```
\a     alarm, that is, the BEL character (hex 07)
\cx    "control-x", where x is any character
\e     escape (hex 1B)
\f     formfeed (hex 0C)
\n     newline (hex 0A)
\r     carriage return (hex 0D)
\t     tab (hex 09)
\xhh   character with hex code hh
\ddd   character with octal code ddd, or backreference
```

The precise effect of "\cx" is as follows: if "x" is a lower case letter, it is converted to upper case. Then bit 6 of the character (hex 40) is inverted. Thus "\cz" becomes hex 1A, but "\c{" becomes hex 3B, while "\c;" becomes hex 7B.

After "\x", up to two hexadecimal digits are read (letters can be in upper or lower case).

After "\0" up to two further octal digits are read. In both cases, if there are fewer than two digits, just those that are present are used. Thus the sequence "\0\x\07" specifies two binary zeros followed by a BEL character. Make sure you supply two digits after the initial zero if the character that follows is itself an octal digit.

The handling of a backslash followed by a digit other than 0 is complicated. Outside a character class, PCRE reads it and any following digits as a decimal number. If the number is less than 10, or if there have been at least that many previous capturing left parentheses in the expression, the entire sequence is taken as a *back reference*. A description of how this works is given later, following the discussion of parenthesized subpatterns.

Inside a character class, or if the decimal number is greater than 9 and there have not been that many capturing subpatterns, PCRE re-reads up to three octal digits following the backslash, and generates a single byte from the least significant 8 bits of the value. Any subsequent digits stand for themselves. For example:

```
\040   is another way of writing a space
\40    is the same, provided there are fewer than 40
         previous capturing subpatterns
\7     is always a back reference
\11    might be a back reference, or another way of
         writing a tab
\011   is always a tab
\0113  is a tab followed by the character "3"
\113   is the character with octal code 113 (since there
         can be no more than 99 back references)
\377   is a byte consisting entirely of 1 bits
\81    is either a back reference, or a binary zero
```

followed by the two characters "8" and "1"

Note that octal values of 100 or greater must not be introduced by a leading zero, because no more than three octal digits are ever read.

All the sequences that define a single byte value can be used both inside and outside character classes. In addition, inside a character class, the sequence "\b" is interpreted as the backspace character (hex 08). Outside a character class it has a different meaning (see below).

The third use of backslash is for specifying generic character types:

```
\d     any decimal digit
\D     any character that is not a decimal digit
\s     any whitespace character
\S     any character that is not a whitespace character
\w     any "word" character
\W     any "non-word" character
```

Each pair of escape sequences partitions the complete set of characters into two disjoint sets. Any given character matches one, and only one, of each pair.

A "word" character is any letter or digit or the underscore character, that is, any character which can be part of a Perl "word". The definition of letters and digits is controlled by PCRE's character tables, and may vary if locale-specific matching is taking place (see "Locale support" above). For example, in the "fr" (French) locale, some character codes greater than 128 are used for accented letters, and these are matched by \w.

These character type sequences can appear both inside and outside character classes. They each match one character of the appropriate type. If the current matching point is at the end of the subject string, all of them fail, since there is no character to match.

The fourth use of backslash is for certain simple assertions. An assertion specifies a condition that has to be met at a particular point in a match, without consuming any characters from the subject string. The use of subpatterns for more complicated assertions is described below. The backslashed assertions are

```
\b     word boundary
\B     not a word boundary
\A     start of subject (independent of multiline mode)
\Z     end of subject or newline at end (independent of
multiline mode)
\z     end of subject (independent of multiline mode)
```

These assertions may not appear in character classes (but note that "\b" has a different meaning, namely the backspace character, inside a character class).

A word boundary is a position in the subject string where

the current character and the previous character do not both match \w or \W (i.e. one matches \w and the other matches \W), or the start or end of the string if the first or last character matches \w, respectively.

The \A, \Z, and \z assertions differ from the traditional circumflex and dollar (described below) in that they only ever match at the very start and end of the subject string, whatever options are set. They are not affected by the PCRE_NOTBOL or PCRE_NOTEOL options. The difference between \Z and \z is that \Z matches before a newline that is the last character of the string as well as at the end of the string, whereas \z matches only at the end.

Cicumflex and dollar

Outside a character class, in the default matching mode, the circumflex character is an assertion which is true only if the current matching point is at the start of the subject string. Inside a character class, circumflex has an entirely different meaning (see below).

Circumflex need not be the first character of the pattern if a number of alternatives are involved, but it should be the first thing in each alternative in which it appears if the pattern is ever to match that branch. If all possible alternatives start with a circumflex, that is, if the pattern is constrained to match only at the start of the subject, it is said to be an "anchored" pattern. (There are also other constructs that can cause a pattern to be anchored.)

A dollar character is an assertion which is true only if the current matching point is at the end of the subject string, or immediately before a newline character that is the last character in the string (by default). Dollar need not be the last character of the pattern if a number of alternatives are involved, but it should be the last item in any branch in which it appears. Dollar has no special meaning in a character class.

The meaning of dollar can be changed so that it matches only at the very end of the string, by setting the PCRE_DOLLAR_ENDONLY option at compile or matching time. This does not affect the \Z assertion.

The meanings of the circumflex and dollar characters are changed if the PCRE_MULTILINE option is set. When this is the case, they match immediately after and immediately before an internal "\n" character, respectively, in addition to matching at the start and end of the subject string. For example, the pattern /^abc$/ matches the subject string "def\nabc" in multiline mode, but not otherwise. Consequently, patterns that are anchored in single line mode because all branches start with "^" are not anchored in multiline mode. The PCRE_DOLLAR_ENDONLY option is ignored if PCRE_MULTILINE is set.

Note that the sequences \A, \Z, and \z can be used to match the start and end of the subject in both modes, and if all

branches of a pattern start with \A is it always anchored, whether PCRE_MULTILINE is set or not.

FULL STOP

Outside a character class, a dot in the pattern matches any one character in the subject, including a non-printing character, but not (by default) newline. If the PCRE_DOTALL option is set, then dots match newlines as well. The handling of dot is entirely independent of the handling of circumflex and dollar, the only relationship being that they both involve newline characters. Dot has no special meaning in a character class.

Square brackets

An opening square bracket introduces a character class, terminated by a closing square bracket. A closing square bracket on its own is not special. If a closing square bracket is required as a member of the class, it should be the first data character in the class (after an initial circumflex, if present) or escaped with a backslash.

A character class matches a single character in the subject; the character must be in the set of characters defined by the class, unless the first character in the class is a circumflex, in which case the subject character must not be in the set defined by the class. If a circumflex is actually required as a member of the class, ensure it is not the first character, or escape it with a backslash.

For example, the character class [aeiou] matches any lower case vowel, while [^aeiou] matches any character that is not a lower case vowel. Note that a circumflex is just a convenient notation for specifying the characters which are in the class by enumerating those that are not. It is not an assertion: it still consumes a character from the subject string, and fails if the current pointer is at the end of the string.

When caseless matching is set, any letters in a class represent both their upper case and lower case versions, so for example, a caseless [aeiou] matches "A" as well as "a", and a caseless [^aeiou] does not match "A", whereas a caseful version would.

The newline character is never treated in any special way in character classes, whatever the setting of the PCRE_DOTALL or PCRE_MULTILINE options is. A class such as [^a] will always match a newline.

The minus (hyphen) character can be used to specify a range of characters in a character class. For example, [d-m] matches any letter between d and m, inclusive. If a minus character is required in a class, it must be escaped with a backslash or appear in a position where it cannot be interpreted as indicating a range, typically as the first or last

character in the class.

It is not possible to have the literal character "]" as the end character of a range. A pattern such as [W-]46] is interpreted as a class of two characters ("W" and "-") followed by a literal string "46]", so it would match "W46]" or "-46]". However, if the "]" is escaped with a backslash it is interpreted as the end of range, so [W-\]46] is interpreted as a single class containing a range followed by two separate characters. The octal or hexadecimal representation of "]" can also be used to end a range.

Ranges operate in ASCII collating sequence. They can also be used for characters specified numerically, for example [\000-\037]. If a range that includes letters is used when caseless matching is set, it matches the letters in either case. For example, [W-c] is equivalent to [][\^_‘wxyzabc], matched caselessly, and if character tables for the "fr" locale are in use, [\xc8-\xcb] matches accented E characters in both cases.

The character types \d, \D, \s, \S, \w, and \W may also appear in a character class, and add the characters that they match to the class. For example, [\dABCDEF] matches any hexadecimal digit. A circumflex can conveniently be used with the upper case character types to specify a more restricted set of characters than the matching lower case type. For example, the class [^\W_] matches any letter or digit, but not underscore.

All non-alphameric characters other than \, -, ^ (at the start) and the terminating] are non-special in character classes, but it does no harm if they are escaped.

Vertical bar

Vertical bar characters are used to separate alternative patterns. For example, the pattern

```
gilbert|sullivan
```

matches either "gilbert" or "sullivan". Any number of alternatives may appear, and an empty alternative is permitted (matching the empty string). The matching process tries each alternative in turn, from left to right, and the first one that succeeds is used. If the alternatives are within a subpattern (defined below), "succeeds" means matching the rest of the main pattern as well as the alternative in the subpattern.

Internal option setting

The settings of PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL, and PCRE_EXTENDED can be changed from within the pattern by a sequence of Perl option letters enclosed between "(?" and ")". The option letters are

i for PCRE_CASELESS
m for PCRE_MULTILINE
s for PCRE_DOTALL
x for PCRE_EXTENDED

For example, (?im) sets caseless, multiline matching. It is also possible to unset these options by preceding the letter with a hyphen, and a combined setting and unsetting such as (?im-sx), which sets PCRE_CASELESS and PCRE_MULTILINE while unsetting PCRE_DOTALL and PCRE_EXTENDED, is also permitted. If a letter appears both before and after the hyphen, the option is unset.

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern (defined below), the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

(?i)abc
a(?i)bc
ab(?i)c
abc(?i)

which in turn is the same as compiling the pattern abc with PCRE_CASELESS set. In other words, such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at top level, the rightmost setting is used.

If an option change occurs inside a subpattern, the effect is different. This is a change of behaviour in Perl 5.005. An option change inside a subpattern affects only that part of the subpattern that follows it, so

(a(?i)b)c

matches abc and aBc and no other strings (assuming PCRE_CASELESS is not used). By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example,

(a(?i)b|c)

matches "ab", "aB", "c", and "C", even though when matching "C" the first branch is abandoned before the option setting. This is because the effects of option settings happen at compile time. There would be some very weird behaviour otherwise.

The PCRE-specific options PCRE_UNGREEDY and PCRE_EXTRA can be changed in the same way as the Perl-compatible options by using the characters U and X respectively. The (?X) flag setting is special in that it must always occur earlier in the pattern than any of the additional features it turns on, even when it is at top level. It is best put at the start.

subpatterns

Subpatterns are delimited by parentheses (round brackets), which can be nested. Marking part of a pattern as a subpattern does two things:

1. It localizes a set of alternatives. For example, the pattern

```
cat(aract|erpillar|)
```

matches one of the words "cat", "cataract", or "caterpillar". Without the parentheses, it would match "cataract", "erpillar" or the empty string.

2. It sets up the subpattern as a capturing subpattern (as defined above). When the whole pattern matches, that portion of the subject string that matched the subpattern is passed back to the caller via the *ovector* argument of **pcre_exec()**. Opening parentheses are counted from left to right (starting from 1) to obtain the numbers of the capturing subpatterns.

For example, if the string "the red king" is matched against the pattern

```
the ((red|white) (king|queen))
```

the captured substrings are "red king", "red", and "king", and are numbered 1, 2, and 3.

The fact that plain parentheses fulfil two functions is not always helpful. There are often times when a grouping subpattern is required without a capturing requirement. If an opening parenthesis is followed by "?:", the subpattern does not do any capturing, and is not counted when computing the number of any subsequent capturing subpatterns. For example, if the string "the white queen" is matched against the pattern

```
the ((?:red|white) (king|queen))
```

the captured substrings are "white queen" and "queen", and are numbered 1 and 2. The maximum number of captured substrings is 99, and the maximum number of all subpatterns, both capturing and non-capturing, is 200.

As a convenient shorthand, if any option settings are required at the start of a non-capturing subpattern, the option letters may appear between the "?" and the ":". Thus the two patterns

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

match exactly the same set of strings. Because alternative branches are tried from left to right, and options are not reset until the end of the subpattern is reached, an option setting in one branch does affect subsequent branches, so the above patterns match "SUNDAY" as well as "Saturday".

Repetition

Repetition is specified by quantifiers, which can follow any of the following items:

a single character, possibly escaped
the . metacharacter
a character class
a back reference (see next section)
a parenthesized subpattern (unless it is an assertion - see below)

The general repetition quantifier specifies a minimum and maximum number of permitted matches, by giving the two numbers in curly brackets (braces), separated by a comma. The numbers must be less than 65536, and the first must be less than or equal to the second. For example:

z{2,4}

matches "zz", "zzz", or "zzzz". A closing brace on its own is not a special character. If the second number is omitted, but the comma is present, there is no upper limit; if the second number and the comma are both omitted, the quantifier specifies an exact number of required matches. Thus

[aeiou]{3,}

matches at least 3 successive vowels, but may match many more, while

\d{8}

matches exactly 8 digits. An opening curly bracket that appears in a position where a quantifier is not allowed, or one that does not match the syntax of a quantifier, is taken as a literal character. For example, {,6} is not a quantifier, but a literal string of four characters.

The quantifier {0} is permitted, causing the expression to behave as if the previous item and the quantifier were not present.

For convenience (and historical compatibility) the three most common quantifiers have single-character abbreviations:

* is equivalent to {0,}
\+ is equivalent to {1,}
? is equivalent to {0,1}

It is possible to construct infinite loops by following a subpattern that can match no characters with a quantifier that has no upper limit, for example:

(a?)*

Earlier versions of Perl and PCRE used to give an error at compile time for such patterns. However, because there are

cases where this can be useful, such patterns are now accepted, but if any repetition of the subpattern does in fact match no characters, the loop is forcibly broken.

By default, the quantifiers are "greedy", that is, they match as much as possible (up to the maximum number of permitted times), without causing the rest of the pattern to fail. The classic example of where this gives problems is in trying to match comments in C programs. These appear between the sequences /* and */ and within the sequence, individual * and / characters may appear. An attempt to match C comments by applying the pattern

```
/\*.*\*/
```

to the string

```
/* first command */  not comment  /* second comment */
```

fails, because it matches the entire string due to the greediness of the .* item.

However, if a quantifier is followed by a question mark, then it ceases to be greedy, and instead matches the minimum number of times possible, so the pattern

```
/\*.*?\*/
```

does the right thing with the C comments. The meaning of the various quantifiers is not otherwise changed, just the preferred number of matches. Do not confuse this use of question mark with its use as a quantifier in its own right. Because it has two uses, it can sometimes appear doubled, as in

```
\d??\d
```

which matches one digit by preference, but can match two if that is the only way the rest of the pattern matches.

If the PCRE_UNGREEDY option is set (an option which is not available in Perl) then the quantifiers are not greedy by default, but individual ones can be made greedy by following them with a question mark. In other words, it inverts the default behaviour.

When a parenthesized subpattern is quantified with a minimum repeat count that is greater than 1 or with a limited maximum, more store is required for the compiled pattern, in proportion to the size of the minimum or maximum.

If a pattern starts with .* or .{0,} and the PCRE_DOTALL option (equivalent to Perl's /s) is set, thus allowing the . to match newlines, then the pattern is implicitly anchored, because whatever follows will be tried against every character position in the subject string, so there is no point in retrying the overall match at any position after the first. PCRE treats such a pattern as though it were preceded by \A. In cases where it is known that the subject string contains no newlines, it is worth setting PCRE_DOTALL when the pat-

tern begins with .* in order to obtain this optimization, or alternatively using ^ to indicate anchoring explicitly.

When a capturing subpattern is repeated, the value captured is the substring that matched the final iteration. For example, after

(tweedle[dume]{3}\s*)+

has matched "tweedledum tweedledee" the value of the captured substring is "tweedledee". However, if there are nested capturing subpatterns, the corresponding captured values may have been set in previous iterations. For example, after

/(a|(b))+/

matches "aba" the value of the second captured substring is "b".

BACK REFERENCES

Outside a character class, a backslash followed by a digit greater than 0 (and possibly further digits) is a back reference to a capturing subpattern earlier (i.e. to its left) in the pattern, provided there have been that many previous capturing left parentheses.

However, if the decimal number following the backslash is less than 10, it is always taken as a back reference, and causes an error only if there are not that many capturing left parentheses in the entire pattern. In other words, the parentheses that are referenced need not be to the left of the reference for numbers less than 10. See the section entitled "Backslash" above for further details of the handling of digits following a backslash.

A back reference matches whatever actually matched the capturing subpattern in the current subject string, rather than anything matching the subpattern itself. So the pattern

(sens|respons)e and \1ibility

matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility". If caseful matching is in force at the time of the back reference, then the case of letters is relevant. For example,

((?i)rah)\s+\1

matches "rah rah" and "RAH RAH", but not "RAH rah", even though the original capturing subpattern is matched caselessly.

There may be more than one back reference to the same subpattern. If a subpattern has not actually been used in a particular match, then any back references to it always fail. For example, the pattern

```
(a|(bc))\2
```

always fails if it starts to match "a" rather than "bc". Because there may be up to 99 back references, all digits following the backslash are taken as part of a potential back reference number. If the pattern continues with a digit character, then some delimiter must be used to terminate the back reference. If the PCRE_EXTENDED option is set, this can be whitespace. Otherwise an empty comment can be used.

A back reference that occurs inside the parentheses to which it refers fails when the subpattern is first used, so, for example, (a\1) never matches. However, such references can be useful inside repeated subpatterns. For example, the pattern

```
(a|b\1)+
```

matches any number of "a"s and also "aba", "ababaa" etc. At each iteration of the subpattern, the back reference matches the character string corresponding to the previous iteration. In order for this to work, the pattern must be such that the first iteration does not need to match the back reference. This can be done using alternation, as in the example above, or by a quantifier with a minimum of zero.

Assertions

An assertion is a test on the characters following or preceding the current matching point that does not actually consume any characters. The simple assertions coded as \b, \B, \A, \Z, \z, ^ and $ are described above. More complicated assertions are coded as subpatterns. There are two kinds: those that look ahead of the current position in the subject string, and those that look behind it.

An assertion subpattern is matched in the normal way, except that it does not cause the current matching position to be changed. Lookahead assertions start with (?= for positive assertions and (?! for negative assertions. For example,

```
\w+(?=;)
```

matches a word followed by a semicolon, but does not include the semicolon in the match, and

```
foo(?!bar)
```

matches any occurrence of "foo" that is not followed by "bar". Note that the apparently similar pattern

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion (?!foo) is always true when the next three characters are "bar". A lookbehind assertion is needed to achieve this effect.

Lookbehind assertions start with (?<= for positive assertions and (?<! for negative assertions. For example,

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=bullock|donkey)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error at compile time. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is an extension compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. Lookbehinds in conjunction with once-only subpatterns can be particularly useful for matching at the ends of strings; an example is given at the end of the section on once-only subpatterns.

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches "foo" preceded by three digits that are not "999". Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, then there is a check that the same three characters are not "999". This pattern does not match "foo" preceded by six characters, the first of which are digits and the last three of which are not "999". For example, it doesn’t match "123abcfoo". A pattern to do that is

```
(?<=\d{3}...)(?<!999)foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three

characters are not "999".

Assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of "baz" that is preceded by "bar" which in turn is not preceded by "foo", while

```
(?<=\d{3}(?!999)...)foo
```

is another pattern which matches "foo" preceded by three digits and any three characters that are not "999".

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Assertions count towards the maximum of 200 parenthesized subpatterns.

Once-only subpatterns

With both maximizing and minimizing repetition, failure of what follows normally causes the repeated item to be re-evaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern \d+foo when applied to the subject line

```
123456bar
```

After matching all 6 digits and then failing to match "foo", the normal action of the matcher is to try again with only 5 digits matching the \d+ item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be re-evaluated in this way, so the matcher would give up immediately on failing to match "foo" the first time. The notation is another kind of special parenthesis, starting with (?> as in this example:

```
(?>\d+)bar
```

This kind of parenthesis "locks up" the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Back-tracking past it to previous items, however, works as nor-mal.

An alternative description is that a subpattern of this type matches the string of characters that an identical standalone pattern would match, if anchored at the current point in the subject string.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both \d+ and \d+? are prepared to adjust the number of digits they match in order to make the rest of the pattern match, (?>\d+) can only match an entire sequence of digits.

This construction can of course contain arbitrarily complicated subpatterns, and it can be nested.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of the subject string. Consider a simple pattern such as

```
abcd$
```

when applied to a long string which does not match. Because matching proceeds from left to right, PCRE will look for each "a" in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as

```
^.*abcd$
```

then the initial .* matches the entire string at first, but when this fails (because there is no following "a"), it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for "a" covers the entire string, from right to left, so we are no better off. However, if the pattern is written as

```
^(?>.*)(?<=abcd)
```

then there can be no backtracking for the .* item; it can match only the entire string. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time indeed. The pattern

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in <>, followed by either ! or ?. When it matches, it runs quickly. However, if it is applied to

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. This is because the string can be divided between the two repeats in a large number of ways, and all have to be tried. (The example used [!?] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.) If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?]
```

sequences of non-digits cannot be broken, and failure happens quickly.

Conditional subpatterns

It is possible to cause the matching process to obey a subpattern conditionally or to choose between two alternative subpatterns, depending on the result of an assertion, or whether a previous capturing subpattern matched or not. The two possible forms of conditional subpattern are

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

If the condition is satisfied, the yes-pattern is used; otherwise the no-pattern (if present) is used. If there are more than two alternatives in the subpattern, a compile-time error occurs.

There are two kinds of condition. If the text between the parentheses consists of a sequence of digits, then the condition is satisfied if the capturing subpattern of that number has previously matched. Consider the following pattern, which contains non-significant white space to make it more readable (assume the PCRE_EXTENDED option) and to divide it into three parts for ease of discussion:

```
( \( )?    [^()]+    (?(1) \) )
```

The first part matches an optional opening parenthesis, and if that character is present, sets it as the first captured substring. The second part matches one or more characters that are not parentheses. The third part is a conditional subpattern that tests whether the first set of parentheses matched or not. If they did, that is, if subject started with an opening parenthesis, the condition is true, and so the yes-pattern is executed and a closing parenthesis is required. Otherwise, since no-pattern is not present, the subpattern matches nothing. In other words, this pattern matches a sequence of non-parentheses, optionally enclosed in parentheses.

If the condition is not a sequence of digits, it must be an assertion. This may be a positive or negative lookahead or lookbehind assertion. Consider this pattern, again containing non-significant white space, and with the two alternatives on the second line:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

The condition is a positive lookahead assertion that matches an optional sequence of non-letters followed by a letter. In other words, it tests for the presence of at least one letter in the subject. If a letter is found, the subject is matched against the first alternative; otherwise it is matched against the second. This pattern matches strings in one of the two forms dd-aaa-dd or dd-dd-dd, where aaa are letters and dd are digits.

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

If the PCRE_EXTENDED option is set, an unescaped # character outside a character class introduces a comment that continues up to the next newline character in the pattern.

Recursive patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (amongst other things). The special item (?R) is provided for the specific case of recursion. This PCRE pattern solves the parentheses problem (assume the PCRE_EXTENDED option is set so that white space is ignored):

```
\( ( (?>[^()]+) | (?R) )* \)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (i.e. a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it is applied to

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

it yields "no match" quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the + and *

repeats can carve up the subject, and all have to be tested before failure can be reported.

The values set for any capturing subpatterns are those from the outermost level of the recursion at which the subpattern value is set. If the pattern above is matched against

```
(ab(cd)ef)
```

the value for the capturing parentheses is "ef", which is the last value taken on at the top level. If additional parentheses are added, giving

```
\( ( ( (?>[^()]+) | (?R) )* ) \)
   ^                  ^
   ^                  ^ then the string they capture
```

is "ab(cd)ef", the contents of the top level parentheses. If there are more than 15 capturing parentheses in a pattern, PCRE has to obtain extra memory to store data during a recursion, which it does by using pcre_malloc, freeing it via pcre_free afterwards. If no memory can be obtained, it saves data for the first 15 capturing parentheses only, as there is no way to give an out-of-memory error from within a recursion.

Performances

Certain items that may appear in patterns are more efficient than others. It is more efficient to use a character class like [aeiou] than a set of alternatives such as (a|e|i|o|u). In general, the simplest construction that provides the required behaviour is usually the most efficient. Jeffrey Friedl's book contains a lot of discussion about optimizing regular expressions for efficient performance.

When a pattern begins with .* and the PCRE_DOTALL option is set, the pattern is implicitly anchored by PCRE, since it can match only at the start of a subject string. However, if PCRE_DOTALL is not set, PCRE cannot make this optimization, because the . metacharacter does not then match a newline, and if the subject string contains newlines, the pattern may match from the character immediately following one of them instead of from the very start. For example, the pattern

```
(.*) second
```

matches the subject "first\nand second" (where \n stands for a newline character) with the first captured substring being "and". In order to do this, PCRE has to retry the match starting after every newline in the subject.

If you are using such a pattern with subject strings that do not contain newlines, the best performance is obtained by setting PCRE_DOTALL, or starting the pattern with ^.* to indicate explicit anchoring. That saves PCRE from having to scan along the subject looking for a newline to restart at.

Beware of patterns that contain nested indefinite repeats. These can take a long time to run when applied to a string

that does not match. Consider the pattern fragment

```
(a+)*
```

This can match "aaaa" in 33 different ways, and this number increases very rapidly as the string gets longer. (The * repeat can match 0, 1, 2, 3, or 4 times, and for each of those cases other than 0, the + repeats can match different numbers of times.) When the remainder of the pattern is such that the entire match is going to fail, PCRE has in principle to try every possible variation, and this can take an extremely long time.

An optimization catches some of the more simple cases such as

```
(a+)*b
```

where a literal character follows. Before embarking on the standard matching procedure, PCRE checks that there is a "b" later in the subject string, and if there is not, it fails the match immediately. However, when there is no following literal this optimization cannot be used. You can see the difference by comparing the behaviour of

```
(a+)*\d
```

with the pattern above. The former gives a failure almost instantly when applied to a whole line of "a" characters, whereas the latter takes an appreciable time with strings longer than about 20 characters.

LXVI. Regular Expression Functions (POSIX Extended)

Regular expressions are used for complex string manipulation in PHP. The functions that support regular expressions are:

- **ereg()**
- **ereg_replace()**
- **eregi()**
- **eregi_replace()**
- **split()**
- **spliti()**

These functions all take a regular expression string as their first argument. PHP uses the POSIX extended regular expressions as defined by POSIX 1003.2. For a full description of POSIX regular expressions see the regex man pages included in the regex directory in the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

Příklad 1. Regular Expression Examples

```
ereg ("abc", $string);
/* Returns true if "abc"
   is found anywhere in $string. */

ereg ("^abc", $string);
/* Returns true if "abc"
   is found at the beginning of $string. */

ereg ("abc$", $string);
/* Returns true if "abc"
   is found at the end of $string. */

eregi ("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);
/* Returns true if client browser
   is Netscape 2, 3 or MSIE 3. */

ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);
/* Places three space separated words
   into $regs[1], $regs[2] and $regs[3]. */

$string = ereg_replace ("^", "<BR>", $string);
/* Put a <BR> tag at the beginning of $string. */

$string = ereg_replace ("$", "<BR>", $string);
/* Put a <BR> tag at the end of $string. */

$string = ereg_replace ("\n", "", $string);
/* Get rid of any newline
   characters in $string. */
```

ereg (PHP 3, PHP 4)

Regular expression match

```
int ereg (string pattern, string string [, array regs])
```

Searches a *string* for matches to the regular expression given in *pattern*.

If matches are found for parenthesized substrings of *pattern* and the function is called with the third argument *regs*, the matches will be stored in the elements of the array *regs*. $regs[1] will contain the substring which starts at the first left parenthesis; $regs[2] will contain the substring starting at the second, and so on. $regs[0] will contain a copy of *string*.

If **ereg()** finds any matches at all, $regs will be filled with exactly ten elements, even though more or fewer than ten parenthesized substrings may actually have matched. This has no effect on **ereg()**'s ability to match more substrings. If no matches are found, $regs will not be altered by **ereg()**.

Searching is case sensitive.

Returns true if a match for *pattern* was found in *string*, or false if no matches were found or an error occurred.

The following code snippet takes a date in ISO format (YYYY-MM-DD) and prints it in DD.MM.YYYY format:

Příklad 1. ereg() Example

```
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

See also **eregi()**, **ereg_replace()**, and **eregi_replace()**.

ereg_replace (PHP 3, PHP 4)

Replace regular expression

```
string ereg_replace (string pattern, string replacement, string string)
```

This function scans *string* for matches to *pattern*, then replaces the matched text with *replacement*.

The modified string is returned. (Which may mean that the original string is returned if there are no matches to be replaced.)

If *pattern* contains parenthesized substrings, *replacement* may contain substrings of the form *digit*, which will be replaced by the text matching the digit'th parenthesized substring; \\0 will produce the entire contents of string. Up to nine substrings may be used. Parentheses may be nested, in which case they are counted by the opening parenthesis.

If no matches are found in *string*, then *string* will be returned unchanged.

For example, the following code snippet prints "This was a test" three times:

Příklad 1. ereg_replace() Example

```
$string = "This is a test";
echo ereg_replace (" is", " was", $string);
echo ereg_replace ("( )is", "\\1was", $string);
echo ereg_replace ("(( )is)", "\\2was", $string);
```

One thing to take note of is that if you use an integer value as the *replacement* parameter, you may not get the results you expect. This is because **ereg_replace()** will interpret the number as the ordinal value of a character, and apply that. For instance:

Příklad 2. ereg_replace() Example

```
<?php
/* This will not work as expected. */
$num = 4;
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has   words.' */

/* This will work. */
$num = '4';
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has 4 words.' */
?>
```

See also **ereg()**, **eregi()**, and **eregi_replace()**.

eregi (PHP 3, PHP 4)

case insensitive regular expression match

```
int eregi (string pattern, string string [, array regs])
```

This function is identical to **ereg()** except that this ignores case distinction when matching alphabetic characters.

See also **ereg()**, **ereg_replace()**, and **eregi_replace()**.

eregi_replace (PHP 3, PHP 4)

replace regular expression case insensitive

```
string eregi_replace (string pattern, string replacement, string string)
```

This function is identical to **ereg_replace()** except that this ignores case distinction when matching alphabetic characters.

See also **ereg()**, **eregi()**, and **ereg_replace()**.

split (PHP 3, PHP 4)

split string into array by regular expression

```
array split (string pattern, string string [, int limit])
```

Returns an array of strings, each of which is a substring of *string* formed by splitting it on boundaries formed by the regular expression *pattern*. If *limit* is set, the returned array will contain a maximum of *limit* elements with the last element containing the whole rest of *string*. If an error occurs, **split()** returns false.

To split off the first four fields from a line from `/etc/passwd`:

Příklad 1. Split() Example

```
$passwd_list = split (":", $passwd_line, 5);
```

To parse a date which may be delimited with slashes, dots, or hyphens:

Příklad 2. Split() Example

```
$date = "04/30/1973";  // Delimiters may be slash, dot, or hyphen
list ($month, $day, $year) = split ('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Note that *`pattern`* is case-sensitive.

Note that if you don't require the power of regular expressions, it is faster to use **explode()**, which doesn't incur the overhead of the regular expression engine.

For users looking for a way to emulate perl's **$chars = split(", $str)** behaviour, please see the examples for **preg_split()**.

Please note that *`pattern`* is a regular expression. If you want to split on any of the characters which are considered special by regular expressions, you'll need to escape them first. If you think **split()** (or any other regex function, for that matter) is doing something weird, please read the file `regex.7`, included in the `regex/` subdirectory of the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

See also: **spliti()**, **explode()**, and **implode()**.

spliti (PHP 4 >= 4.0.1)

Split string into array by regular expression case insensitive

```
array spliti (string pattern, string string [, int limit])
```

This function is identical to **split()** except that this ignores case distinction when matching alphabetic characters.

See also : **split()**, **explode()** and **implode()**.

sql_regcase (PHP 3, PHP 4)

Make regular expression for case insensitive match

```
string sql_regcase (string string)
```

Returns a valid regular expression which will match *`string`*, ignoring case. This expression is *`string`* with each character converted to a bracket expression; this bracket expression contains that character's uppercase and lowercase form if applicable, otherwise it contains the original character twice.

Příklad 1. Sql_regcase() Example

```
echo sql_regcase ("Foo bar");
```

prints

```
[Ff][Oo][Oo] [Bb][Aa][Rr]
```

.

This can be used to achieve case insensitive pattern matching in products which support only case sensitive regular expressions.

LXVII. Satellite CORBA klient extenze

Satellite extenze se používá pro přístup ke CORBA objektům. Vyžaduje idl_directory= položku v php.ini ukazující na cestu kde jsou všechny používané IDL soubory.

OrbitObject (unknown)

Zpřístupnit CORBA objekty

```
new OrbitObject (string ior)
```

Tato třída poskytuje přístup ke CORBA objektu. Argument *ior* by měl být řetězec obsahující IOR (Interoperable Object Reference) která identifikuje vzdálený objekt.

Příklad 1. Ukázkový IDL soubor

```
interface MyInterface {
    void SetInfo (string info);
    string GetInfo();

    attribute int value;
}
```

Příklad 2. PHP kód pro přístup k MyInterface

```
<?php
$obj = new OrbitObject ($ior);

$obj->SetInfo ("A 2GooD object");

echo $obj->GetInfo();

$obj->value = 42;

echo $obj->value;
?>
```

OrbitEnum (unknown)

Použít CORBA enumerace

```
new OrbitEnum (string id)
```

Tato třída představuje enumeraci určenou argumentem *id* parameter. *id* může být buď název této enumerace (např. "MyEnum"), nebo plné repository id (např. "IDL:MyEnum:1.0").

Příklad 1. Ukázkový IDL soubor

```
enum MyEnum {
    a,b,c,d,e
};
```

Příklad 2. PHP kód pro přístup k MyEnum

```
<?php
$enum = new OrbitEnum ("MyEnum");
```

```
echo $enum->a; /* write 0 */
echo $enum->c; /* write 2 */
echo $enum->e; /* write 4 */
?>
```

OrbitStruct (unknown)

Použít CORBA struktury

```
new OrbitStruct (string id)
```

Tato třída představuje strukturu určenou argumentem *id*. *id* může být buď název této struktury (např. "MyStruct"), nebo plné repository id (např. "IDL:MyStruct:1.0").

Příklad 1. Ukázkový IDL soubor

```
struct MyStruct {
    short shortvalue;
    string stringvalue;
};

interface SomeInterface {
  void SetValues (MyStruct values);
  MyStruct GetValues();
}
```

Příklad 2. PHP kód pro přístup k MyStruct

```
<?php
$obj = new OrbitObject ($ior);

$initial_values = new OrbitStruct ("IDL:MyStruct:1.0");
$initial_values->shortvalue = 42;
$initial_values->stringvalue = "HGTTG";

$obj->SetValues ($initial_values);

$values = $obj->GetValues();

echo $values->shortvalue;
echo $values->stringvalue;
?>
```

satellite_caught_exception (PHP 4 >= 4.0.3)

Zjistit, jestli byla v předchozí funkci zachycena výjimka

```
bool satellite_caught_exception  ()
```

Tato funkce vrací `true`, pokud byla zachycena výjimka.

Příklad 1. Ukázkový IDL soubor

```
/* ++?????++ Out of Cheese Error. Redo From Start. */
exception OutOfCheeseError {
    int parameter;
}

interface AnotherInterface {
    void AskWhy() raises (OutOfCheeseError);
}
```

Příklad 2. PHP kód pro zpracování CORBA výjimek

```
<?php
$obj = new OrbitObject ($ior);

$obj->AskWhy();

if (satellite_caught_exception()) {
    if ("IDL:OutOfCheeseError:1.0" == satellite_exception_id()) {
        $exception = satellite_exception_value();
        echo $exception->parameter;
    }
}
?>
```

satellite_exception_id (PHP 4 >= 4.0.3)

Zjistit repository id poslední výjimky

```
string satellite_exception_id ()
```

Vrací řetězec obsahující repository id (napr. "IDL:MyException:1.0".). Ukázkové použití viz **satellite_caught_exception()**.

satellite_exception_value (PHP 4 >= 4.0.3)

Získat strukturu poslední výjimky

```
OrbitStruct satellite_exception_value  ()
```

Vrátí strukturu výjimky. Ukázkové použití viz **satellite_caught_exception()**.

LXVIII. Funkce pro práci se semafory a sdílenou pamětí

Tato extenze poskytuje semaforové funkce využívající System V semafory. Semafory se dají používat k poskytování exkluzivního přístupu k prostředkům na daném systému, nebo k omezení počtu procesů, které mohou současně používat určitý prostředek.

Tato extenze také poskytuje funkce pro práci se sdílenou pamětí využívající System V sdílenou paměť. Sdílená pmět se dá používat k poskytování přístupu ke globálním proměnným. Různí httpd-daemoni a dokonce i jiné programy (např. Perl, C, ...) mohou k těmto datům přistupovat, a vytvořit tak globální výměnu dat. Pamatujte, že sdílená paměť *není* chráněna proti simultáním přístupům. K synchronizaci použijte semafory.

Tabulka 1. Omezení sdílené paměti systémem Unix

SHMMAX	max. velikost sdílené paměti, normálně 131072 bytů
SHMMIN	min. velikost sdílené paměti, normálne 1 byte
SHMMNI	max. počet segmentů sdílené paměti, normálně 100
SHMSEG	max. počet segmentů sdílené paměti na proces, normálně 6

Poznámka: Tyto funkce nefungují na Windows.

sem_get (PHP 3>= 3.0.6, PHP 4)

Získat id semaforu

```
int sem_get (int key [, int max_acquire [, int perm]])
```

Vrací idenfifikátor semaforu nebo `false`.

sem_get() vrací id, které se dá použít k přístupu k System V semaforu s daným klíčem. Podle potřeby se vytvoří nový semafor s přístupovými právy definovanými v *`perm`* (default je 0666). Počet procesů, které mohou tento semafor získat současně je *`max_acquire`* (default je 1). Tato hodnota je ale nastavena pouze pokud tento proces zjistí, že k tomuto semaforu není současně připojen jiný proces.

Druhé volání **sem_get()** se stejným *`key`* vrátí jiný identifikátor semaforu, ale oba identifikátory ukazují na stejný semafor.

Viz také: **sem_acquire()** a **sem_release()**.

> **Poznámka:** Tato funkce nefunguje na Windows.

sem_acquire (PHP 3>= 3.0.6, PHP 4)

Získat semafor

```
int sem_acquire (int sem_identifier)
```

Při úspěchu vrací `true`, při chybě `false`.

sem_acquire() blokuje (pokud je optřeba) až do získání semaforu. Proces pokoušející se získat semafor, který už získal bude blokovat navěky, pokud by získání tohoto semaforu způsobilo překročení jeho hodnoty max_acquire.

Po zpracování požadavku se všechny získané, ale explicitně neuvolněné semafoty uvolní automaticky, a vygeneruje se varování.

Viz také: **sem_get()** a **sem_release()**.

sem_release (PHP 3>= 3.0.6, PHP 4)

Uvolnit semafor

```
int sem_release (int sem_identifier)
```

Při úspěchu vrací `true`, jinak `false`.

sem_release() uvolňí semafor, pokud ho volající proces drží, jinak se vygeneruje varování.

Po uvolnění může být semafor znovu získán pomocí **sem_acquire()**.

Viz také: **sem_get()** a **sem_acquire()**.

> **Poznámka:** Tato funkce nefunguje na Windows.

shm_attach (PHP 3>= 3.0.6, PHP 4)

Vytvořit nebo otevřít segment sdílené paměti

```
int shm_attach (int key [, int memsize [, int perm]])
```

shm_attach() vrací id, které se dá použít k přístupu k System V sdílené paměti s daným klíčem; první volání vytvoří segment paměti o velikosti mem_size (default: sysvshm.init_mem v konfiguračním souboru, jinak 10000 bytů) a s volitelnými právy (default: 0666).

Druhé volání **shm_attach()** se stejným *key* vrátí jiný identifikáto, ale oba ukazují na stejnou sdílenou paměť. *memsize* a *perm* se v takovém případě ignorují.

Poznámka: Tato funkce nefunguje na Windows.

shm_detach (PHP 3>= 3.0.6, PHP 4)

Odpojit se od segmentu sdílené paměti

```
int shm_detach (int shm_identifier)
```

shm_detach() odpojuje od sdílené paměti identifikované *shm_identifier* vytvořeným **shm_attach()**. Pamatujte, že tato sdílená paměť dál existuje a drží si data.

shm_remove (PHP 3>= 3.0.6, PHP 4)

Odstranit sdílenou paměť ze systému

```
int shm_remove (int shm_identifier)
```

Odstraní sdílenou paměť z UNIXového systému. Všechna data v ní budou zničena.

Poznámka: Tato funkce nefunguje na Windows.

shm_put_var (PHP 3>= 3.0.6, PHP 4)

Vložit nebo modifikovat proměnnou do sdílené paměti

```
int shm_put_var (int shm_identifier, int variable_key, mixed variable)
```

Vloží nebo modifikuje *variable* s daným *variable_key*. Všechny typy proměnných (double, int, string, array) jsou podporovány.

Poznámka: Tato funkce nefunguje na Windows.

shm_get_var (PHP 3>= 3.0.6, PHP 4)

Vrátit proměnnou ze sdílené paměti

```
mixed shm_get_var (int id, int variable_key)
```

shm_get_var() vrací proměnnou s daným *variable_key*. Proměnná zůstává ve sdílené paměti.

Poznámka: Tato funkce nefunguje na Windows.

shm_remove_var (PHP 3>= 3.0.6, PHP 4)

Odstranit proměnnou ze sdílené paměti

```
int shm_remove_var (int id, int variable_key)
```

Odstraní proměnnou s daným*variable_key* a uvolní zabranou paměť.

Poznámka: Tato funkce nefunguje na Windows.

LXIX. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Configuration notes: There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.

Tabulka 1. SESAM Configuration directives

Directive	Meaning
`php3_sesam_oml`	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

`php3_sesam_configfile`	Name of SESAM application configuration file. Required for using SESAM functions. Example: `php3_sesam_configfile $SESAM.SESAM.CONF.AW` It will usually contain a configuration like (see SESAM reference manual):
	`CNF=B`
	`NAM=K`
	`NOTYPE`

`php3_sesam_messagecatalog`	Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the `CNF` and `NAM` parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web

server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives `php3_magic_quotes_gpc` and `php3_magic_quotes_sybase` to `On` for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**. When fetching a row using a "scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Tabulka 2. Scrolled Cursor Post-Processing

Scroll Type	Action
`SESAM_SEEK_NEXT`	none
`SESAM_SEEK_PRIOR`	none
`SESAM_SEEK_FIRST`	set scroll type to `SESAM_SEEK_NEXT`
`SESAM_SEEK_LAST`	set scroll type to `SESAM_SEEK_PRIOR`
`SESAM_SEEK_ABSOLUTE`	Auto-Increment internal offset value
`SESAM_SEEK_RELATIVE`	none. (maintain global default *`offset value, which allows for, e.g., fetching each 10th row backwards)`*

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with **sesam_fetch_row()**, the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL

security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., `TEXT` could be replaced by `VARCHAR(max. size)`).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of `int(4)` (intended use: integers up to '9999'), SESAM requires simply `int` for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: `DATE`, `TIME(3)` and `TIMESTAMP(3)`.

- SQL types with vendor-specific `unsigned`, `zerofill`, or `auto_increment` attributes

  `Unsigned` and `zerofill` are not supported. `Auto_increment` is automatic (use `"INSERT ... VALUES(*, ...)"` instead of `"... VALUES(0, ...)"` to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT '0000'**

  Numeric variables must not be initialized with string constants. Use **DEFAULT 0** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in:
  ```
  CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL );
  ```

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword `RESTRICT` or `CASCADE`. When specifying `RESTRICT`, an error is returned if there are dependent objects (e.g., VIEWs), while with `CASCADE`, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Tabulka 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	"integer"
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double"
DATE, TIME, TIMESTAMP	"string"

SQL Type	PHP Type
VARCHAR, CHARACTER	"string"

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use **isset()** or **empty()** to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Příklad 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Příklad 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) or the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm), both available online, or use the respective manuals.

sesam_connect (PHP 3 CVS only)

Open SESAM database connection

```
boolean sesam_connect (string catalog, string schema, string user)
```

Returns TRUE if a connection to the SESAM database was made, or FALSE on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistant" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also **sesam_disconnect()**.

Příklad 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM";
?>
```

sesam_disconnect (PHP 3 CVS only)

Detach from SESAM connection

```
boolean sesam_disconnect(void);
```

Returns: always TRUE.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit **sesam_rollback()** is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: **sesam_connect()**.

Příklad 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_settransaction (PHP 3 CVS only)

Set SESAM transaction parameters

```
boolean sesam_settransaction (int isolation_level, int read_only)
```

Returns: TRUE if the values are valid, and the **settransaction()** operation was successful, FALSE otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Tabulka 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Tabulka 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Příklad 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

sesam_commit (PHP 3 CVS only)

Commit pending updates to the SESAM database

```
boolean sesam_commit(void);
```

Returns: TRUE on success, FALSE on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling **sesam_disconnect()**) will be discarded by an implied **sesam_rollback()** call.

See also: **sesam_rollback()**.

Příklad 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_rollback (PHP 3 CVS only)

Discard any pending updates to the SESAM database

```
boolean sesam_rollback(void);
```

Returns: TRUE on success, FALSE on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the **sesam_disconnect()** function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: **sesam_commit()**.

Příklad 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_execimm (PHP 3 CVS only)

Execute an "immediate" SQL-statement

```
string sesam_execimm (string query)
```

Returns: A SESAM "result identifier" on success, or FALSE on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the **sesam_affected_rows()** function.

Note that **sesam_query()** can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return `$err["sqlstate"] == "42SBW"`.

The returned "result identifier" can not be used for retrieving anything but the **sesam_affected_rows()**; it is only returned for symmetry with the **sesam_query()** function.

```
$stmt = "INSERT INTO mytable VALUES (’one’, ’two’)";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: **sesam_query()** and **sesam_affected_rows()**.

sesam_query (PHP 3 CVS only)

Perform a SESAM SQL query and prepare the result

```
string sesam_query (string query [, boolean scrollable])
```

Returns: A SESAM "result identifier" on success, or `FALSE` on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent **sesam_fetch_row()** or **sesam_fetch_result()** call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *`scrollable`* parameter) cursor will be allocated. If *`scrollable`* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**.

For "immediate" statements, the number of affected rows is saved for retrieval by the **sesam_affected_rows()** function.

See also: **sesam_fetch_row()** and **sesam_fetch_result()**.

Příklad 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
        /* Span the table head over SESAM’s "Multiple Fields": */
        if ($colattr["count"] > 1) {
            echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                "(1..".$colattr["count"].")</TH>\n";
            $col += $colattr["count"] - 1;
```

```
        } else
            echo "  <TH>" . $colattr["name"] . "</TH>\n";
    }
    echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result ($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
            } else {
                echo "  <TD>-empty-</TD>\n";
            }
        }
        echo " </TR>\n";
    }
}
while ($ok["truncated"]) { // while there may be more data
    echo "</TABLE>\n";
}
// free result id
sesam_free_result($result);
?>
```

sesam_num_fields (PHP 3 CVS only)

Return the number of fields/columns in a result set

```
int sesam_num_fields (string result_id)
```

After calling **sesam_query()** with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or FALSE on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: **sesam_query()** and **sesam_field_array()** for a way to distinguish between "multiple field" columns and regular columns.

sesam_field_name (PHP 3 CVS only)

Return one column name of the result set

```
int sesam_field_name (string result_id, int index)
```

Returns the name of a field (i.e., the column name) in the result set, or FALSE on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

Poznámka: The column index is zero-based, not one-based as in SESAM.

See also: **sesam_field_array()**. It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_diagnostic (PHP 3 CVS only)

Return status information for last SESAM call

```
array sesam_diagnostic(void);
```

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Tabulka 1. Status information returned by sesam_diagnostic()

Element	Contents
$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Příklad 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        -$colspan;
    if ($err["errcol"] == -1)
        -$colspan;
    if ($err["rowcount"] == 0)
        -$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
   htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
```

```
                if (! ($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
                print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
            }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: **sesam_errormsg()** for simple access to the error string only

sesam_fetch_result (PHP 3 CVS only)

Return all or part of a query result

```
mixed sesam_fetch_result (string result_id [, int max_rows])
```

Returns a mixed array with the query result entries, optionally limited to a maximum of *max_rows* rows. Note that both row and column indexes are zero-based.

Tabulka 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr["count"]	number of columns in result set (or zero if this was an "immediate" query)
int $arr["rows"]	number of rows in result set (between zero and *max_rows)*

Array Element	Contents
boolean $arr["truncated"]	TRUE if the number of rows was at least *max_rows, FALSE otherwise. Note that even when this is TRUE, the next* ***sesam_fetch_result() call may return zero rows because there are no more result entries.***
mixed $arr[col][row]	result data for all the fields at row(row) and column(col), (where the integer index row is between 0 and $arr["rows"]-1, and col is between 0 and $arr["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php **isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.**

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: **sesam_fetch_row()**, and **sesam_field_array()** to check for "multiple fields". See the description of the **sesam_query()** function for a complete example using **sesam_fetch_result()**.

sesam_affected_rows (PHP 3 CVS only)

Get number of rows affected by an immediate query

```
int sesam_affected_rows (string result_id)
```

result_id is a valid result id returned by **sesam_query()**.

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like INSERT, UPDATE and DELETE) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: **sesam_query()** and **sesam_execimm()**

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_errormsg (PHP 3 CVS only)

Returns error message of last SESAM call

```
string sesam_errormsg(void);
```

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: **sesam_diagnostic()** for the full set of SESAM SQL status information

sesam_field_array (PHP 3 CVS only)

Return meta information about individual columns in a result

```
array sesam_field_array (string result_id)
```

result_id is a valid result id returned by **sesam_query()**.

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Tabulka 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr[col]["name"]	column name for column(col), where col is between 0 and $arr["count"]-1. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.
string $arr[col]["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the **sesam_query() description for a sample use of the "count" attribute.**
string $arr[col]["type"]	php variable type of the data for column(col), where col is between 0 and $arr["count"]-1. The returned value can be one of • "integer" • "double" • "string" depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.

Array Element	Contents
string $arr[col]["sqltype"]	SQL variable type of the column data for column(col), where col is between 0 and $arr["count"]-1. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.
string $arr[col]["length"]	The SQL "length" attribute of the SQL variable in column(col), where col is between 0 and $arr["count"]-1. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.
string $arr[col]["precision"]	The "precision" attribute of the SQL variable in column(col), where col is between 0 and $arr["count"]-1. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.
string $arr[col]["scale"]	The "scale" attribute of the SQL variable in column(col), where col is between 0 and $arr["count"]-1. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the **sesam_query()** function for an example of the **sesam_field_array()** use.

sesam_fetch_row (PHP 3 CVS only)

Fetch one row as an array

```
array sesam_fetch_row (string result_id [, int whence [, int offset]])
```

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the **count()** function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Tabulka 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as *offset (Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented)*
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where *offset can be a positive or negative offset value.*

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by **sesam_seek_row()**) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between 0 and $array["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php **isset()** function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Příklad 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_array()** which returns an associative array, and **sesam_fetch_result()** which returns many rows per invocation.

sesam_fetch_array (PHP 3 CVS only)

Fetch one row as an associative array

```
array sesam_fetch_array (string result_id [, int whence [, int offset]])
```

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

sesam_fetch_array() is an alternative version of **sesam_fetch_row()**. Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

For the valid values of the optional *whence* and *offset* parameters, see the **sesam_fetch_row()** function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call **sesam_fetch_row()** or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be "multi(1)", "multi(2)", and "multi(3)" respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Příklad 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_row()** which returns an indexed array.

sesam_seek_row (PHP 3 CVS only)

Set scrollable cursor mode for subsequent fetches

```
boolean sesam_seek_row (string result_id, int whence [, int offset])
```

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling **sesam_query()**).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Tabulka 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as *offset (Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented)*

Value	Constant	Meaning
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where *offset can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).*

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_free_result (PHP 3 CVS only)

Releases resources for the query

```
int sesam_free_result (string result_id)
```

Releases resources for the query associated with *result_id*. Returns FALSE on error.

LXX. Session handling functions

Session support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site.

If you are familiar with the session management of PHPLIB, you will notice that some concepts are similar to PHP's session support.

A visitor accessing your web site is assigned an unique id, the so-called session id. This is either stored in a cookie on the user side or is propagated in the URL.

The session support allows you to register arbitrary numbers of variables to be preserved across requests. When a visitor accesses your site, PHP will check automatically (if session.auto_start is set to 1) or on your request (explicitly through **session_start()** or implicitly through **session_register()**) whether a specific session id has been sent with the request. If this is the case, the prior saved environment is recreated.

All registered variables are serialized after the request finishes. Registered variables which are undefined are marked as being not defined. On subsequent accesses, these are not defined by the session module unless the user defines them later.

The `track_vars` and `register_globals` configuration settings influence how the session variables get stored and restored.

> **Poznámka:** As of PHP 4.0.3, `track_vars` is always turned on.

If `track_vars` is enabled and `register_globals` is disabled, only members of the global associative array $HTTP_SESSION_VARS can be registered as session variables. The restored session variables will only be available in the array $HTTP_SESSION_VARS.

Příklad 1. Registering a variable with `track_vars` enabled

```
<?php
session_register("count");
$HTTP_SESSION_VARS["count"]++;
?>
```

If `register_globals` is enabled, then all global variables can be registered as session variables and the session variables will be restored to corresponding global variables.

Příklad 2. Registering a variable with `register_globals` enabled

```
<?php
session_register("count");
$count++;
?>
```

If both `track_vars` and `register_globals` are enabled, then the globals variables and the $HTTP_SESSION_VARS entries will reference the same value.

There are two methods to propagate a session id:

- Cookies
- URL parameter

The session module supports both methods. Cookies are optimal, but since they are not reliable (clients are not bound to accept them), we cannot rely on them. The second method embeds the session id directly into URLs.

PHP is capable of doing this transparently when compiled with `-enable-trans-sid`. If you enable this option, relative URIs will be changed to contain the session id automatically. Alternatively, you can use the constant `SID`

which is defined, if the client did not send the appropriate cookie. `SID` is either of the form `session_name=session_id` or is an empty string.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

Příklad 3. Counting the number of hits of a single user

```
<?php
session_register ("count");
$count++;
?>

Hello visitor, you have seen this page <?php echo $count; ?> times.<p>

<php?
# the <?=SID?> is necessary to preserve the session id
# in the case that the user has disabled cookies
?>

To continue, <A HREF="nextpage.php?<?=SID?>">click here</A>
```

The `<?=SID?>` is not necessary, if `-enable-trans-sid` was used to compile PHP.

> **Poznámka:** Non-relative URLs are assumed to point to external sites and hence don't append the SID, as it would be a security risk to leak the SID to an different server.

To implement database storage, or any other storage method, you will need to use **session_set_save_handler()** to create a set of user-level storage functions.

The session management system supports a number of configuration options which you can place in your php.ini file. We will give a short overview.

- `session.save_handler` defines the name of the handler which is used for storing and retrieving data associated with a session. Defaults to `files`.
- `session.save_path` defines the argument which is passed to the save handler. If you choose the default files handler, this is the path where the files are created. Defaults to `/tmp`.

> **Varování**
>
> If you leave this set to a world-readable directory, such as `/tmp` (the default), other users on the server may be able to hijack sessions by getting the list of files in that directory.

- `session.name` specifies the name of the session which is used as cookie name. It should only contain alphanumeric characters. Defaults to `PHPSESSID`.
- `session.auto_start` specifies whether the session module starts a session automatically on request startup. Defaults to `0` (disabled).
- `session.cookie_lifetime` specifies the lifetime of the cookie in seconds which is sent to the browser. The value 0 means "until the browser is closed." Defaults to `0`.
- `session.serialize_handler` defines the name of the handler which is used to serialize/deserialize data. Currently, a PHP internal format (name `php`) and WDDX is supported (name `wddx`). WDDX is only available, if

PHP is compiled with WDDX support. Defaults to `php`.

- `session.gc_probability` specifies the probability that the gc (garbage collection) routine is started on each request in percent. Defaults to `1`.
- `session.gc_maxlifetime` specifies the number of seconds after which data will be seen as 'garbage' and cleaned up.
- `session.referer_check` contains the substring you want to check each HTTP Referer for. If the Referer was sent by the client and the substring was not found, the embedded session id will be marked as invalid. Defaults to the empty string.
- `session.entropy_file` gives a path to an external resource (file) which will be used as an additional entropy source in the session id creation process. Examples are `/dev/random` or `/dev/urandom` which are available on many Unix systems.
- `session.entropy_length` specifies the number of bytes which will be read from the file specified above. Defaults to `0` (disabled).
- `session.use_cookies` specifies whether the module will use cookies to store the session id on the client side. Defaults to `1` (enabled).
- `session.cookie_path` specifies path to set in session_cookie. Defaults to `/`.
- `session.cookie_domain` specifies domain to set in session_cookie. Default is none at all.
- `session.cache_limiter` specifies cache control method to use for session pages (nocache/private/public). Defaults to `nocache`.
- `session.cache_expire` specifies time-to-live for cached session pages in minutes, this has no effect for nocache limiter. Defaults to `180`.

Poznámka: Session handling was added in PHP 4.0.

session_start (PHP 4)

Initialize session data

```
bool session_start(void);
```

session_start() creates a session (or resumes the current one based on the session id being passed via a GET variable or a cookie).

This function always returns true.

Poznámka: This function was added in PHP 4.0.

session_destroy (PHP 4)

Destroys all data registered to a session

```
bool session_destroy(void);
```

session_destroy() destroys all of the data associated with the current session.

This function returns true on success and false on failure to destroy the session data.

session_name (PHP 4)

Get and/or set the current session name

```
string session_name ([string name])
```

session_name() returns the name of the current session. If *name* is specified, the name of the current session is changed to its value.

The session name references the session id in cookies and URLs. It should contain only alphanumeric characters; it should be short and descriptive (i.e. for users with enabled cookie warnings). The session name is reset to the default value stored in `session.name` at request startup time. Thus, you need to call **session_name()** for every request (and before **session_start()** or **session_register()** are called).

Příklad 1. session_name() examples

```
<?php

# set the session name to WebsiteID

$previous_name = session_name ("WebsiteID");

echo "The previous session name was $previous_name<p>";
?>
```

Poznámka: This function was added in PHP 4.0.

session_module_name (PHP 4)

Get and/or set the current session module

```
string session_module_name ([string module])
```

session_module_name() returns the name of the current session module. If *module* is specified, that module will be used instead.

Poznámka: This function was added in PHP 4.0.

session_save_path (PHP 4)

Get and/or set the current session save path

```
string session_save_path ([string path])
```

session_save_path() returns the path of the current directory used to save session data. If *path* is specified, the path to which data is saved will be changed.

Poznámka: On some operating systems, you may want to specify a path on a filesystem that handles lots of small files efficiently. For example, on Linux, reiserfs may provide better performance than ext2fs.

Poznámka: This function was added in PHP 4.0.

session_id (PHP 4)

Get and/or set the current session id

```
string session_id ([string id])
```

session_id() returns the session id for the current session. If *id* is specified, it will replace the current session id.

The constant SID can also be used to retrieve the current name and session id as a string suitable for adding to URLs.

session_register (PHP 4)

Register one or more variables with the current session

```
bool session_register (mixed name [, mixed ...])
```

session_register() variable number of arguments, any of which can be either a string holding the variable name or an array consisting of such variable names or other arrays. For each encountered variable name, **session_register()** registers the global variable named by it with the current session.

This function returns true when the variable is successfully registered with the session.

Poznámka: This function was added in PHP 4.0.

session_unregister (PHP 4)

Unregister a variable from the current session

```
bool session_unregister (string name)
```

session_unregister() unregisters (forgets) the global variable named *name* from the current session.

This function returns true when the variable is successfully unregistered from the session.

Poznámka: This function was added in PHP 4.0.

session_unset (PHP 4 >= 4.0b4)

Free all session variables

```
void session_unset(void);
```

The **session_unset()** function free's all session variables currently registered.

session_is_registered (PHP 4)

Find out if a variable is registered in a session

```
bool session_is_registered (string name)
```

session_is_registered() returns true if there is a variable with the name *name* registered in the current session.

Poznámka: This function was added in PHP 4.0.

session_get_cookie_params (PHP 4 >= 4.0RC2)

Get the session cookie parameters

```
array session_get_cookie_params (void);
```

The **session_get_cookie_params()** function returns an array with the current session cookie information, the array contains the following items:

- "lifetime" - The lifetime of the cookie.
- "path" - The path where information is stored.

- "domain" - The domain of the cookie.

session_set_cookie_params (PHP 4 >= 4.0b4)

Set the session cookie parameters

```
void session_set_cookie_params (int lifetime [, string path [, string domain]])
```

Set cookie parameters defined in the php.ini file. The effect of this function only lasts for the duration of the script.

session_decode (PHP 4)

Decodes session data from a string

```
bool session_decode (string data)
```

session_decode() decodes the session data in *data*, setting variables stored in the session.

Poznámka: This function was added in PHP 4.0.

session_encode (PHP 4)

Encodes the current session data as a string

```
string session_encode(void);
```

session_encode() returns a string with the contents of the current session encoded within.

Poznámka: This function was added in PHP 4.0.

session_set_save_handler (PHP 4 >= 4.0b4)

Sets user-level session storage functions

```
void session_set_save_handler  (string open, string close, string read, string write,
string destroy, string gc)
```

session_set_save_handler() sets the user-level session storage functions which are used for storing and retrieving data associated with a session. This is most useful when a storage method other than those supplied by PHP sessions is preferred. i.e. Storing the session data in a local database.

Poznámka: You must set the configuration option *session.save_handler* to *user* in your php.ini file for **session_set_save_handler()** to take effect.

> **Poznámka:** The "write" handler is not executed until after the output stream is closed. Thus, output from debugging statements in the "write" handler will never be seen in the browser. If debugging output is necessary, it is suggested that the debug output be written to a file instead.

The following example provides file based session storage similar to the PHP sessions default save handler *files*. This example could easily be extended to cover database storage using your favorite PHP supported database engine.

Příklad 1. session_set_save_handler() example

```
<?php

function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function close() {
  return(true);
}

function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return("");
  }

}

function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destroy ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/*****************************************
 * WARNING - You will need to implement some *
 * sort of garbage collection routine here.  *
 *****************************************/
function gc ($maxlifetime) {
  return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");
```

```
session_start();

// proceed to use sessions normally

?>
```

session_cache_limiter (PHP 4 >= 4.0.3)

Get and/or set the current cache limiter

```
string session_cache_limiter ([string cache_limiter])
```

session_cache_limiter() returns the name of the current cache limiter. If *cache_limiter* is specified, the name of the current cache limiter is changed to the new value.

The cache limiter controls the cache control HTTP headers sent to the client. These headers determine the rules by which the page content may be cached. Setting the cache limiter to `nocache`, for example, would disallow any client-side caching. A value of `public`, however, would permit caching. It can also be set to `private`, which is slightly more restrictive than `public`.

The cache limiter is reset to the default value stored in `session.cache_limiter` at request startup time. Thus, you need to call **session_cache_limiter()** for every request (and before **session_start()** is called).

Příklad 1. session_cache_limiter() examples

```
<?php

# set the cache limiter to 'private'

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "The cache limiter is now set to $cache_limiter<p>";
?>
```

Poznámka: This function was added in PHP 4.0.3.

LXXI. Funkce pro práci se sdílenou pamětí

Shmop snadno použitelná sada funkcí, která PHP umožňuje číst, zapisovat, vytvářet a mazat segmenty UNIXové sdílené paměti. Tyto funkce na Windows nefungují, protože tento systém nepodporuje sdílenou paměť. Pokud chcete shmop používat, budete muset PHP zkompilovat s `-enable-shmop`.

> **Poznámka:** Názvy funkcí popisovaných v této kapitole začínají v PHP 4.0.3 na **shm_()**, ale od PHP 4.0.4 se jejich názvy změnily na **shmop_()**.

Příklad 1. Přehled operací se sdílenou pamětí

```
<?php

// Vytvořit 100 bytový blok sdílené paměti se system id 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Nepodařilo se vytvořit segment sdílené paměti\n";
}

// Zjistit velikost bloku sdílené paměti
$shm_size = shmop_size($shm_id);
echo "SHM blok o velikosti : ".$shm_size. " byl vytvořen.\n";

// Zapíšeme do sdílené paměti zkušební řetězec
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("můj blok sdílené paměti")) {
echo "Nepodařilo se zapsat kompletní data\n";
}

// Načteme řetězec zpátky
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Nepodařilo se číst z bloku sdílené paměti\n";
}
echo "Data ve sdílené paměti byla: ".$my_string."\n";

//Smažeme tento blok a zavřeme segment sdílené paměti
if(!shmop_delete($shm_id)) {
echo "Nepodařilo se smazat blok sdílené paměti.";
}
shmop_close($shm_id);

?>
```

shmop_open (PHP 4 >= 4.0.4)

Vytvořit nebo otevřít blok sdílené paměti

```
int shmop_open (int key, string flags, int mode, int size)
```

shmop_open() vytvoří nebo otevře blok sdílené paměti.

shmop_open() přijímá 4 argumenty: klíč, což je system id bloku sdílené paměti; tento argument může být předán jako desítkové nebo hexadecimální číslo. Druhý argument jsou parametry:

- "a" pro přístup (nastavuje IPC_EXCL) tento parametr použijte, pokud chcete otevřít existující segment sdílené paměti
- "c" pro vytvoření (nastavuje IPC_CREATE) tento parametr použijte, pokud chcete vytvořit nový segment sdílené paměti

  Třetí argument je mód, což jsou přístupová práva, která chcete tomuto segmentu přiřadit; jsou stejná jako práva pro soubory. Přístupová práva musí být předána jako oktalové číslo, např. 0644. Poslední argument je velikost bloku sdílené paměti, který chcete vytvořit, v bytech.

  Poznámka: Pozn.: Pokud otvíráte existující segment paměti, 3. 4. argument by měly být předány jako 0. Při úspěchu **shmop_open()** vrací id, které můžete použít k přístupu na tento segment sdílené paměti.

Příklad 1. Vytvoření bloku sdílené paměti

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

Tato ukázka otevřela blok sdílené paměti se system id 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Přečíst data z bloku sdílené paměti

```
string shmop_read (int shmid, int start, int count)
```

shmop_read() čte řetězec z bloku sdílené paměti.

shmop_read() přijímá 3 argumenty: *`shmid`*, což je id bloku sdílené paměti vytvořeného funkcí **shmop_open()**, *`start`*, což je offset na kterém má čtení začít, a *`count`*, což je počet bytů, které se mají přečíst.

Příklad 1. Čtení bloku sdílené paměti

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

Tato ukázka přečte z bloku sdílené paměti 50 bytů a umístí načtená data do `$shm_data`.

shmop_write (PHP 4 >= 4.0.4)

Zapsat data do bloku sdílené paměti

```
int shmop_write (int shmid, string data, int offset)
```

shmop_write() zapíše řetězec do bloku sdílené paměti.

shmop_write() přijímá 3 argumenty: *`shmid`*, což je id bloku sdílené paměti vytvořeného funkcí **shmop_open()**, *`data`*, což je řetězec, který se zapíše do bloku sdílené paměti, a *`offset`*, což je offset na kterém má zápis začít.

Příklad 1. Zápis do bloku sdílené paměti

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

Tato ukázka zapíše data obsažená v `$my_string` do bloku sdílené paměti, a `$shm_bytes_written` obsahuje počet zapsaných bytů.

shmop_size (PHP 4 >= 4.0.4)

Zjistit velikost bloku sdílené paměti

```
int shmop_size (int shmid)
```

shmop_size() se používá ke zjištění velikosti bloku sdílené paměti v bytech.

shmop_size() přijímá *`shmid`*, což je idenfifikátor vytvořený funkcí **shmop_open()**. **shmop_size()** vrací integer představující počet bytů, které blok sdílené paměti zabírá.

Příklad 1. Zjištění velikosti bloku sdílené paměti

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

Tato ukázka zapíše velikost bloku sdílené paměti určeného identifikátorem `$shm_id` do `$shm_size`.

shmop_delete (PHP 4 >= 4.0.4)

Smazat blok sdílené paměti

```
int shmop_delete (int shmid)
```

shmop_delete() se používá ke smazání bloku sdílené paměti.

shmop_delete() přijímá *`shmid`*, což je identifikátor bloku sdílené paměti vytvořený funkcí **shmop_open()**. Při úspěchu vrací 1, jinak 0.

Příklad 1. Smazání bloku sdílené paměti

```
<?php
```

```
shmop_delete($shm_id);
?>
```

Tato ukázka smaže blok sdílené paměti pojmenovaný `$shm_id`.

shmop_close (PHP 4 >= 4.0.4)

Zavřít blok sdílené paměti

```
int shmop_close (int shmid)
```

shmop_close() se používá k zavření bloku sdílené paměti.

shmop_close() přijímá *`shmid`*, což je identifikátor bloku sdílené paměti vytvořený funkcí **shmop_open()**.

Příklad 1. Zavření bloku sdílené paměti

```
<?php
shmop_close($shm_id);
?>
```

Tato ukázka zavře blok sdílené paměti pojmenovaný `$shm_id`.

LXXII. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at http://reality.sgi.com/grafica/flash/. Once you have libswf all you need to do is to configure `-with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Příklad 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

It will produce the animation found at the following url (http://www.designmultimedia.com/swfphp/test.swf).

Poznámka: SWF support was added in PHP 4 RC2.

The libswf does not have support for Windows. The development of that library has been stopped, and the source is not available to port it to another systems.

swf_openfile (PHP 4 >= 4.0RC2)

Open a new Shockwave Flash file

```
void swf_openfile  (string filename, float width, float height, float framerate, float
r, float g, float b)
```

The **swf_openfile()** function opens a new file named *filename* with a width of *width* and a height of *height* a frame rate of *framerate* and background with a red color of *r* a green color of *g* and a blue color of *b*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_closefile (PHP 4 >= 4.0RC2)

Close the current Shockwave Flash file

```
void swf_closefile  ([int return_file])
```

Close a file that was opened by the **swf_openfile()** function. If the *return_file* parameter is set then the contents of the SWF file are returned from the function.

Příklad 1. Creating a simple flash file based on user input and outputting it and saving it in a database

```
<?php

// The $text variable is submitted by the
// user

// Global variables for database
// access (used in the swf_savedata() function)
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";

swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);

    swf_definetext (11, $text, 1);

    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
    swf_popmatrix ();

    swf_showframe ();

    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();

$data = swf_closefile (1);

$data ?
  swf_savedata ($data) :
  die ("Error could not save SWF file");

// void swf_savedata (string data)
```

```
// Save the generated file a database
// for later retrieval
function swf_savedata ($data)
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;

    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";

    $sth = @mysql_query ($stmt, $dbh);

    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
>
```

swf_labelframe (PHP 4 >= 4.0RC2)

Label the current frame

```
void swf_labelframe  (string name)
```

Label the current frame with the name given by the *name* parameter.

swf_showframe (PHP 4 >= 4.0RC2)

Display the current frame

```
void swf_showframe (void);
```

The swf_showframe function will output the current frame.

swf_setframe (PHP 4 >= 4.0RC2)

Switch to a specified frame

```
void swf_setframe  (int framenumber)
```

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_getframe (PHP 4 >= 4.0RC2)

Get the frame number of the current frame

```
int swf_getframe (void);
```

The **swf_getframe()** function gets the number of the current frame.

swf_mulcolor (PHP 4 >= 4.0RC2)

Sets the global multiply color to the rgba value specified

```
void swf_mulcolor  (float r, float g, float b, float a)
```

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

Poznámka: The *rgba* values can be either positive or negative.

swf_addcolor (PHP 4 >= 4.0RC2)

Set the global add color to the rgba value specified

```
void swf_addcolor  (float r, float g, float b, float a)
```

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Poznámka: The *rgba* values can be either positive or negative.

swf_placeobject (PHP 4 >= 4.0RC2)

Place an object onto the screen

```
void swf_placeobject  (int objid, int depth)
```

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by **swf_mulcolor()**) and the current addcolor (specified by **swf_addcolor()**) to color the object and it uses the current matrix to position the object.

Poznámka: Full RGBA colors are supported.

swf_modifyobject (PHP 4 >= 4.0RC2)

Modify an object

```
void swf_modifyobject  (int depth, int how)
```

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function **swf_mulcolor()**) and addcolor (specified by the function **swf_addcolor()**) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_removeobject (PHP 4 >= 4.0RC2)

Remove an object

```
void swf_removeobject  (int depth)
```

Removes the object at the depth specified by *depth*.

swf_nextid (PHP 4 >= 4.0RC2)

Returns the next free object id

```
int swf_nextid (void);
```

The **swf_nextid()** function returns the next available object id.

swf_startdoaction (PHP 4 >= 4.0RC2)

Start a description of an action list for the current frame

```
void swf_startdoaction (void);
```

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_actiongotoframe (PHP 4 >= 4.0RC2)

Play a frame and then stop

```
void swf_actiongotoframe  (int framenumber)
```

The **swf_actionGotoFrame()** function will go to the frame specified by *framenumber*, play it, and then stop.

swf_actiongeturl (PHP 4 >= 4.0RC2)

Get a URL from a Shockwave Flash movie

```
void swf_actiongeturl  (string url, string target)
```

The **swf_actionGetUrl()** function gets the URL specified by the parameter *url* with the target *target*.

swf_actionnextframe (PHP 4 >= 4.0RC2)

Go foward one frame

```
void swf_actionnextframe (void);
```

Go foward one frame.

swf_actionprevframe (PHP 4 >= 4.0RC2)

Go backwards one frame

```
void swf_actionprevframe (void);
```

swf_actionplay (PHP 4 >= 4.0RC2)

Start playing the flash movie from the current frame

```
void swf_actionplay (void);
```

Start playing the flash movie from the current frame.

swf_actionstop (PHP 4 >= 4.0RC2)

Stop playing the flash movie at the current frame

```
void swf_actionstop (void);
```

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4 >= 4.0RC2)

Toggle between low and high quality

```
void swf_actiontogglequality (void);
```

Toggle the flash movie between high and low quality.

swf_actionwaitforframe (PHP 4 >= 4.0RC2)

Skip actions if a frame has not been loaded

```
void swf_actionwaitforframe  (int framenumber, int skipcount)
```

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *framenumber* parameter has been loaded, if not it will skip the number of actions specified by the *skipcount* parameter. This can be useful for "Loading..." type animations.

swf_actionsettarget (PHP 4 >= 4.0RC2)

Set the context for actions

```
void swf_actionsettarget  (string target)
```

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actiongotolabel (PHP 4 >= 4.0RC2)

Display a frame with the specified label

```
void swf_actiongotolabel  (string label)
```

The **swf_actionGotoLabel()** function displays the frame with the label given by the *label* parameter and then stops.

swf_enddoaction (PHP 4 >= 4.0RC2)

End the current action

```
void swf_enddoaction (void);
```

Ends the current action started by the **swf_startdoaction()** function.

swf_defineline (PHP 4 >= 4.0RC2)

Define a line

```
void swf_defineline  (int objid, float x1, float y1, float x2, float y2, float width)
```

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definerect (PHP 4 >= 4.0RC2)

Define a rectangle

```
void swf_definerect  (int objid, float x1, float y1, float x2, float y2, float width)
```

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definepoly (PHP 4 >= 4.0.0)

Define a polygon

```
void swf_definepoly  (int objid, array coords, int npoints, float width)
```

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_startshape (PHP 4 >= 4.0RC2)

Start a complex shape

```
void swf_startshape  (int objid)
```

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_shapelinesolid (PHP 4 >= 4.0RC2)

Set the current line style

```
void swf_shapelinesolid  (float r, float g, float b, float a, float width)
```

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapefilloff (PHP 4 >= 4.0RC2)

Turns off filling

```
void swf_shapefilloff (void);
```

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4 >= 4.0RC2)

Set the current fill style to the specified color

```
void swf_shapefillsolid  (float r, float g, float b, float a)
```

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapefillbitmapclip (PHP 4 >= 4.0RC2)

Set current fill mode to clipped bitmap

```
void swf_shapefillbitmapclip  (int bitmapid)
```

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4 >= 4.0RC2)

Set current fill mode to tiled bitmap

```
void swf_shapefillbitmaptile  (int bitmapid)
```

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapemoveto (PHP 4 >= 4.0RC2)

Move the current position

```
void swf_shapemoveto  (float x, float y)
```

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_shapelineto (PHP 4 >= 4.0RC2)

Draw a line

```
void swf_shapelineto  (float x, float y)
```

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapecurveto (PHP 4 >= 4.0RC2)

Draw a quadratic bezier curve between two points

```
void swf_shapecurveto  (float x1, float y1, float x2, float y2)
```

The **swf_shapecurveto()** function draws a quadratic bezier curve from the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapecurveto3 (PHP 4 >= 4.0RC2)

Draw a cubic bezier curve

```
void swf_shapecurveto3  (float x1, float y1, float x2, float y2, float x3, float y3)
```

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapearc (PHP 4 >= 4.0RC2)

Draw a circular arc

```
void swf_shapearc  (float x, float y, float r, float ang1, float ang2)
```

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_endshape (PHP 4 >= 4.0RC2)

Completes the definition of the current shape

```
void swf_endshape (void);
```

The **swf_endshape()** completes the definition of the current shape.

swf_definefont (PHP 4 >= 4.0RC2)

Defines a font

```
void swf_definefont  (int fontid, string fontname)
```

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_setfont (PHP 4 >= 4.0RC2)

Change the current font

```
void swf_setfont  (int fontid)
```

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_fontsize (PHP 4 >= 4.0RC2)

Change the font size

```
void swf_fontsize  (float size)
```

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant (PHP 4 >= 4.0RC2)

Set the font slant

```
void swf_fontslant  (float slant)
```

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4 >= 4.0RC2)

Set the current font tracking

```
void swf_fonttracking  (float tracking)
```

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getfontinfo (PHP 4 >= 4.0RC2)

The height in pixels of a capital A and a lowercase x

```
array swf_getfontinfo (void);
```

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_definetext (PHP 4 >= 4.0RC2)

Define a text string

```
void swf_definetext  (int objid, string str, int docenter)
```

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_textwidth (PHP 4 >= 4.0RC2)

Get the width of a string

```
float swf_textwidth  (string str)
```

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_definebitmap (PHP 4 >= 4.0RC2)

Define a bitmap

```
void swf_definebitmap  (int objid, string image_name)
```

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_getbitmapinfo (PHP 4 >= 4.0RC2)

Get information about a bitmap

```
array swf_getbitmapinfo  (int bitmapid)
```

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_startsymbol (PHP 4 >= 4.0RC2)

Define a symbol

```
void swf_startsymbol  (int objid)
```

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_endsymbol (PHP 4 >= 4.0RC2)

End the definition of a symbol

```
void swf_endsymbol (void);
```

The **swf_endsymbol()** function ends the definition of a symbol that was started by the **swf_startsymbol()** function.

swf_startbutton (PHP 4 >= 4.0RC2)

Start the definition of a button

```
void swf_startbutton  (int objid, int type)
```

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel

from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_addbuttonrecord (PHP 4 >= 4.0RC2)

Controls location, appearance and active area of the current button

```
void swf_addbuttonrecord  (int states, int shapeid, int depth)
```

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Příklad 1. Swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_oncondition (PHP 4 >= 4.0RC2)

Describe a transition used to trigger an action list

```
void swf_oncondition  (int transition)
```

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown

- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_endbutton (PHP 4 >= 4.0RC2)

End the definition of the current button

```
void swf_endbutton (void);
```

The **swf_endButton()** function ends the definition of the current button.

swf_viewport (PHP 4 >= 4.0RC2)

Select an area for future drawing

```
void swf_viewport  (double xmin, double xmax, double ymin, double ymax)
```

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

swf_ortho (PHP 4 >= 4.0.1)

Defines an orthographic mapping of user coordinates onto the current viewport

```
void swf_ortho  (double xmin, double xmax, double ymin, double ymax, double zmin,
double zmax)
```

The **swf_ortho()** funcion defines a orthographic mapping of user coordinates onto the current viewport.

swf_ortho2 (PHP 4 >= 4.0RC2)

Defines 2D orthographic mapping of user coordinates onto the current viewport

```
void swf_ortho2  (double xmin, double xmax, double ymin, double ymax)
```

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_perspective (PHP 4 >= 4.0RC2)

Define a perspective projection transformation

```
void swf_perspective  (double fovy, double aspect, double near, double far)
```

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

> **Poznámka:** Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_polarview (PHP 4 >= 4.0RC2)

Define the viewer's position with polar coordinates

```
void swf_polarview  (double dist, double azimuth, double incidence, double twist)
```

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_lookat (PHP 4 >= 4.0RC2)

Define a viewing transformation

```
void swf_lookat  (double view_x, double view_y, double view_z, double reference_x,
double reference_y, double reference_z, double twist)
```

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_pushmatrix (PHP 4 >= 4.0RC2)

Push the current transformation matrix back unto the stack

```
void swf_pushmatrix (void);
```

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_popmatrix (PHP 4 >= 4.0RC2)

Restore a previous transformation matrix

```
void swf_popmatrix (void);
```

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_scale (PHP 4 >= 4.0RC2)

Scale the current transformation

```
void swf_scale  (double x, double y, double z)
```

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_translate (PHP 4 >= 4.0RC2)

Translate the current transformations

```
void swf_translate  (double x, double y, double z)
```

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_rotate (PHP 4 >= 4.0RC2)

Rotate the current transformation

```
void swf_rotate  (double angle, string axis)
```

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_posround (PHP 4 >= 4.0RC2)

Enables or Disables the rounding of the translation when objects are placed or moved

```
void swf_posround  (int round)
```

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

LXXIII. SNMP functions

In order to use the SNMP functions on Unix you need to install the UCD SNMP (http://ucd-snmp.ucdavis.edu/) package. On Windows these functions are only available on NT and not on Win95/98.

Important: In order to use the UCD SNMP package, you need to define NO_ZEROLENGTH_COMMUNITY to 1 before compiling it. After configuring UCD SNMP, edit config.h and search for NO_ZEROLENGTH_COMMUNITY. Uncomment the #define line. It should look like this afterwards:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

If you see strange segmentation faults in combination with SNMP commands, you did not follow the above instructions. If you do not want to recompile UCD SNMP, you can compile PHP with the –enable-ucd-snmp-hack switch which will work around the misfeature.

snmpget (PHP 3, PHP 4)

Fetch an SNMP object

```
string snmpget (string hostname, string community, string object_id [, int timeout [,
int retries]])
```

Returns SNMP object value on success and false on error.

The **snmpget()** function is used to read the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
```

snmpset (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Set an SNMP object

```
bool snmpset (string hostname, string community, string object_id, string type, mixed
value [, int timeout [, int retries]])
```

Sets the specified SNMP object value, returning true on success and false on error.

The **snmpset()** function is used to set the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

snmpwalk (PHP 3, PHP 4)

Fetch all the SNMP objects from an agent

```
array snmpwalk (string hostname, string community, string object_id [, int timeout [,
int retries]])
```

Returns an array of SNMP object values starting from the **object_id()** as root and false on error.

snmpwalk() function is used to read all the values from an SNMP agent specified by the *hostname*. *Community* specifies the read community for that agent. A null *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for ($i=0; $i<count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Query for a tree of information about a network entity

```
array snmpwalkoid (string hostname, string community, string object_id [, int timeout
[, int retries]])
```

Returns an associative array with object ids and their respective object value starting from the *object_id* as root and false on error.

snmpwalkoid() function is used to read all object ids and their respective values from an SNMP agent specified by the hostname. Community specifies the read *community* for that agent. A null *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

The existence of **snmpwalkoid()** and **snmpwalk()** has historical reasons. Both functions are provided for backward compatibility.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Fetch the current value of the UCD library's quick_print setting

```
boolean snmp_get_quick_print (void )
```

Returns the current value stored in the UCD Library for quick_print. quick_print is off by default.

```
$quickprint = snmp_get_quick_print();
```

Above function call would return false if quick_print is on, and true if quick_print is on.

snmp_get_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

See: **snmp_set_quick_print()** for a full description of what quick_print does.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Set the value of quick_print within the UCD SNMP library.

```
void snmp_set_quick_print (boolean quick_print)
```

Sets the value of quick_print within the UCD SNMP library. When this is set (1), the SNMP library will return 'quick printed' values. This means that just the value will be printed. When quick_print is not enabled (default) the UCD

SNMP library prints extra information including the type of the value (i.e. IpAddress or OID). Additionally, if quick_print is not enabled, the library prints additional hex values for all strings of three characters or less.

Setting quick_print is often used when using the information returned rather then displaying it.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

The first value printed might be: 'Timeticks: (0) 0:00:00.00', whereas with quick_print enabled, just '0:00:00.00' would be printed.

By default the UCD SNMP library returns verbose values, quick_print is used to return only the value.

Currently strings are still returned with extra quotes, this will be corrected in a later release.

snmp_set_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

LXXIV. Socket functions

The socket extension implements a low-level interface to the socket communication functions, providing the possibility to act as a socket server as well as a client.

The socket functions described here are part of an extension to PHP which must be enabled at compile time by giving the `-enable-sockets` option to **configure**.

For a more generic client-side socket interface, see **fsockopen()** and **pfsockopen()**.

When using these functions, it is important to remember that while many of them have identical names to their C counterparts, they often have different declarations. Please be sure to read the descriptions to avoid confusion.

That said, those unfamiliar with socket programming can still find a lot of useful material in the appropriate Unix man pages, and there is a great deal of tutorial information on socket programming in C on the web, much of which can be applied, with slight modifications, to socket programming in PHP.

Příklad 1. Socket example: Simple TCP/IP server

This example shows a simple talkback server. Change the `address` and `port` variables to suit your setup and execute. You may then connect to the server with a command similar to: **telnet 192.168.1.53 10000** (where the address and port match your setup). Anything you type will then be output on the server side, and echoed back to you. To disconnect, enter 'quit'.

```
<?php
error_reporting (E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit (0);

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() failed: reason: " . strerror ($sock) . "\n";
}

if (($ret = bind ($sock, $address, $port)) < 0) {
    echo "bind() failed: reason: " . strerror ($ret) . "\n";
}

if (($ret = listen ($sock, 5)) < 0) {
    echo "listen() failed: reason: " . strerror ($ret) . "\n";
}

do {
    if (($msgsock = accept_connect($sock)) < 0) {
        echo "accept_connect() failed: reason: " . strerror ($msgsock) . "\n";
        break;
    }
    do {
        $buf = '';
        $ret = read ($msgsock, $buf, 2048);
        if ($ret < 0) {
            echo "read() failed: reason: " . strerror ($ret) . "\n";
            break 2;
        }
        if ($ret == 0) {
            break 2;
        }
        $buf = trim ($buf);
        if ($buf == 'quit') {
            close ($msgsock);
            break 2;
        }
        $talkback = "PHP: You said '$buf'.\n";
        write ($msgsock, $talkback, strlen ($talkback));
```

```
        echo "$buf\n";
    } while (true);
    close ($msgsock);
} while (true);

close ($sock);
?>
```

Příklad 2. Socket example: Simple TCP/IP client

This example shows a simple, one-shot HTTP client. It simply connects to a page, submits a HEAD request, echoes the reply, and exits.

```
<?php
error_reporting (E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname (’www’, ’tcp’);

/* Get the IP address for the target host. */
$address = gethostbyname (’www.php.net’);

/* Create a TCP/IP socket. */
$socket = socket (AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket() failed: reason: " . strerror ($socket) . "\n";
} else {
    "socket() successful: " . strerror ($socket) . "\n";
}

echo "Attempting to connect to ’$address’ on port ’$service_port’...";
$result = connect ($socket, $address, $service_port);
if ($result < 0) {
    echo "connect() failed.\nReason: ($result) " . strerror($result) . "\n";
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = ”;

echo "Sending HTTP HEAD request...";
write ($socket, $in, strlen ($in));
echo "OK.\n";

echo "Reading response:\n\n";
while (read ($socket, $out, 2048)) {
    echo $out;
}

echo "Closing socket...";
close ($socket);
echo "OK.\n\n";
?>
```

accept_connect (PHP 4 >= 4.0.2)

Accepts a connection on a socket

```
int accept_connect (int socket)
```

After the socket *socket* has been created using **socket()**, bound to a name with **bind()**, and told to listen for connections with **listen()**, this function will accept incoming connections on that socket. Once a successful connection is made, a new socket descriptor is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **accept_connect()** will block until a connection becomes present. If *socket* has been made non-blocking using **socket_set_blocking()** or **set_nonblock()**, an error code will be returned.

The socket descriptor returned by **accept_connect()** may not be used to accept new connections. The original listening socket *socket*, however, remains open and may be reused.

Returns a new socket descriptor on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **bind()**, **connect()**, **listen()**, **socket()**, **socket_get_status()**, and **strerror()**.

bind (PHP 4 >= 4.0.2)

Binds a name to a socket

```
int bind (int socket, string address [, int port])
```

bind() binds the name given in *address* to the socket described by *socket*, which must be a valid socket descriptor created with **socket()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the `AF_INET` family; or the pathname of a Unix-domain socket, if the socket family is `AF_UNIX`.

The *port* parameter is only used when connecting to an `AF_INET` socket, and designates the port on the remote host to which a connection should be made.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **accept_connect()**, **connect()**, **listen()**, **socket()**, **socket_get_status()**, and **strerror()**.

close (PHP 4 >= 4.0.2)

Closes a file descriptor

```
bool close (int socket)
```

close() closes the file (or socket) descriptor given by *socket*.

Note that **close()** should not be used on PHP file descriptors created with **fopen()**, **popen()**, **fsockopen()**, or **psockopen()**; it is meant for sockets created with **socket()** or **accept_connect()**.

Returns true on success, or false if an error occurs (i.e., *socket* is invalid).

See also **bind()**, **listen()**, **socket()**, **socket_get_status()**, and **strerror()**.

connect (PHP 4 >= 4.0.2)

Initiates a connection on a socket

```
int connect (int socket, string address [, int port])
```

Initiates a connection using the socket descriptor *socket*, which must be a valid socket descriptor created with **socket()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. 127.0.0.1), if the socket is of the AF_INET family; or the pathname of a Unix-domain socket, if the socket family is AF_UNIX.

The *port* parameter is only used when connecting to an AF_INET socket, and designates the port on the remote host to which a connection should be made.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **bind()**, **listen()**, **socket()**, **socket_get_status()**, and **strerror()**.

listen (PHP 4 >= 4.0.2)

Listens for a connection on a socket

```
int listen (int socket, int backlog)
```

After the socket *socket* has been created using **socket()** and bound to a name with **bind()**, it may be told to listen for incoming connections on *socket*. A maximum of *backlog* incoming connections will be queued for processing.

listen() is applicable only to sockets with type SOCK_STREAM or SOCK_SEQPACKET.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **accept_connect()**, **bind()**, **connect()**, **socket()**, **socket_get_status()**, and **strerror()**.

read (PHP 4 >= 4.0.2)

Read from a socket

```
int read (int socket_des, string &buffer, int length)
```

The function **read()** reads from socket *socket_des*created by the **accept_connect()** function into *&buffer* the number of bytes set by *length*. Otherwise you can use \n, \t or \0 to end reading. Returns number of bytes that have been read.

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, **strerror()**, **socket_get_status()**. and **write()**.

socket (PHP 4 >= 4.0.2)

Create a socket (endpoint for communication)

```
int socket (int domain, int type, int protocol)
```

Creates a communication endpoint (a socket), and returns a descriptor to the socket.

The *domain* parameter sets the domain. Currently, AF_INET and AF_UNIX are understood.

The *type* parameter selects the socket type. This is one of SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, SOCK_RAW, SOCK_RDM, or SOCK_PACKET.

protocol sets the protocol.

Returns a valid socket descriptor on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

For more information on the usage of **socket()**, as well as on the meanings of the various parameters, see the Unix man page socket (2).

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, **strerror()**, and **socket_get_status()**.

strerror (PHP 4 >= 4.0.2)

Return a string describing a socket error

```
string strerror (int errno)
```

strerror() takes as its *errno* parameter the return value of one of the socket functions, and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **strerror()**, and it tells you what happened.

Příklad 1. strerror() example

```
<?php
if (($socket = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
   echo "socket() failed: reason: " . strerror ($socket) . "\n";
}

if (($ret = bind ($socket, '127.0.0.1', 80)) < 0) {
   echo "bind() failed: reason: " . strerror ($ret) . "\n";
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
        bind() failed: reason: Permission denied
```

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, **socket()**, and **socket_get_status()**.

write (PHP 4 >= 4.0.2)

Write to a socket

```
int write (int socket_des, string &buffer, int length)
```

The function **write()** writes to the socket *socket_des* from *&buffer* the number of bytes set by *length*.

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, **read()**, **strerror()**, and **socket_get_status()**.

LXXV. Funkce pro práci s řetězci

Všechny tyto funkce různými způsoby pracují s řetězci. Některé specializovanější funkce najdete v sekcích regulárních výrazů a manipulace s URL.

Informace o chování řetězců, zvláště v souvislosti s použitím jednoduchých uvozovek, dvojitých uvozovek a escape sekvencí viz položku Řetězce v sekci Typy tohoto manuálu.

addcslashes (PHP 4 >= 4.0b4)

Opatřit řetězec lomítky ve stylu jazyka C

```
string addcslashes (string str, string charlist)
```

Vrací řetězec se zpětnými lomítky před znaky, které jsou vypsány v argumentu *charlist*. XXX Excapes \n, \r atd. podobně jako v C, znaky s ASCII kódem nižším než 32 a vyšším než 126 se převedou na osmičkovou reprezentaci. V *charlist* můžete udat rozsah, např. "\0..\37", což by XXX escapovao všechny znaky s ASCII kódem mezi 0 a 31.

Příklad 1. Ukázka addcslashes()

```
$escaped = addcslashes ($not_escaped, "\0..\37!@\177..\377");
```

Poznámka: Přidáno v PHP4b3-dev.

Viz také: **stripcslashes()**, **stripslashes()**, **htmlspecialchars()**, **htmlspecialchars()** a **quotemeta()**.

addslashes (PHP 3, PHP 4)

Opatřit řetězec lomítky

```
string addslashes (string str)
```

Vrací řetězec se zpětnými lomítky před znaky, které potřebují XXX be quoted v databázových dotazech apod. Tyto znaky jsou jednoduchá uvozovka (’), dvojitá uvozovka ("), zpětné lomítko (\) a NUL (null byte).

Viz také: **stripslashes()**, **htmlspecialchars()** a **quotemeta()**.

bin2hex (PHP 3>= 3.0.9, PHP 4)

Převést binární data na hexadecimální reprezentaci

```
string bin2hex (string str)
```

Vrací ASCII řetěec obsahující hexadecimální reprezentaci *str*. Konverze probíhá po bytech, horní slabika první.

Chop (PHP 3, PHP 4)

Odstranit netisknutelné znaky z konce řetězce

```
string chop (string str)
```

Vrací předaný řetězec bez netisknutelných znaků (vč. konců řádku) na konci.

Příklad 1. Ukázka chop()

```
$trimmed = chop ($line);
```

Poznámka: chop() se liší do Perlovské funkce *chop()*, která z řetězce odstraňuje poslední znak.

Viz také: **trim()**, **ltrim()**, **rtrim()** a **chop()**.

Chr (PHP 3, PHP 4)

Vrátit určitý znak

```
string chr (int ascii)
```

Vrací řetězec jednoho znaku obsahující znak specifikovaný argumentem *ascii*.

Příklad 1. Ukázka chr()

```
$str .= chr (27); /* přidá escape znak na konec $str */

/* Toto je většinou užitečnější */

$str = sprintf ("Řetězec konči escape znakem: %c", 27);
```

Tato funkce se doplňuje s funkcí **ord()**. Viz také: **sprintf()** s formátovacím řetězcem %c.

chunk_split (PHP 3>= 3.0.6, PHP 4)

Rozdělit řetězec na menší části

```
string chunk_split (string string [, int chunklen [, string end]])
```

Dá se použít k rozdělení řetězce na menší části, což může být užitečné např. při uvádění výstupu z base64_encode do souladu se sémantikou RFC 2045. Každých *chunklen* (defaultně 76) znaků vloží řetězec *end* (defaultně "\r\n"). Vrací nový řetězec, původní zůstává beze změny.

Příklad 1. Ukázka chunk_split()

```
# naformátuje $data s použitím RFC 2045 sémantiky

$new_string = chunk_split (base64_encode($data));
```

Tato funkce je výrazně rychlejší než **ereg_replace()**.

Poznámka: Tato funkce byla přidána v 3.0.6.

convert_cyr_string (PHP 3>= 3.0.6, PHP 4)

Převést z jedné znakové sady Azbuky do jiné

```
string convert_cyr_string (string str, string from, string to)
```

Tato funkce převede daný řetězec z jedné znakové sady azbuky do jiné. Argumenty *from* a *to* jsou jednotlivé znaky, které představují zdrojovou a cílovou znakovou sadu Azbuky. Podporované typy jsou:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4 >= 4.0b4)

Vrátit informace o znacích použitých v řetězci

```
mixed count_chars (string string [, mode])
```

Počítá počet výskytů všech byte hodnot (0..255) v *string* a vrací je různými způsoby. Volitelný argument *Mode* má defaultní hodnotu 0. V závislosti na *mode* vrací **count_chars()** jednu z následujících možností:

- 0 - pole s klíči tvořenými byte hodnotami a hodnotami tvořenými četností každého bytu.
- 1 - stejné jako 0, ale vrací pouze byte hodnoty s četností vyšší než nula.
- 2 - stejné jako 0, ale vrací pouze byte hodnoty s četností rovnou nule.
- 3 - vrací řetězec obsahující všechny použité byte hodnoty.
- 4 - vrací řetězec obsahující všechny nepoužité byte hodnoty.

Poznámka: Tato funkce byla přidána v PHP 4.0.

crc32 (PHP 4 >= 4.0.1)

Spočítat crc32 XXX polynomial řetězce

```
int crc32 (string str)
```

Generuje 32bitový XXX polynomial kontrolního součtu pro *str*. Obvykle se používá ke kontrole integrity přenášených dat.

Viz také: **md5()**.

crypt (PHP 3, PHP 4)

Zašifrovat řetězec algoritmem DES

```
string crypt (string str [, string salt])
```

crypt() zašifruje řetězec pomocí standardní Unixovské šifrovací metody DES. Argumenty jsou řetězec k zašifrování a volitelný dvouznakový XXX salt, na kterém se šifrování založí. Více informací viz Unixovská man stránka vaší crypt funkce.

Pokud není poskytnut XXX salt argument, PHP jej náhodně vygeneruje.

Některé operační systémy podporují více typů šifrování. Někdy se standardní DES šifrování nahrazuje šifrovacím algoritmem založeným na MD5. Typ šifrování se zvolí podle XXX salt argumentu. Při instalaci PHP zjistí schopnosti funkce crypt a XXX bude přijímat XXX salt pro jiné typy šifrování. Při absenci XXX salt PHP defaultně automaticky vygeneruje standardní dvouznakový DES XXX salt, nicméně pokud je defaultním typem šifrování na daném systému MD5, vygeneruje náhodný XXX salt kompatibilní s MD5. PHP vytváří konstantu CRYPT_SALT_LENGTH, která vám řekne, jestli se na váš systém hodí běžný dvouznakový XXX salt nebo delší dvanáctiznakový MD5 XXX salt.

Pokud používáte poskytnutý XXX salt, měli byste si být vědomi toho, že se generuje jednou. Pokud tuto funkci voláte rekurzivně, může to mít účinek na vzhled, a, do určité míry, bezpečnost.

U standardního DES šifrování **crypt()** přidá XXX salt jako první dva znaky výstupu.

Na systémech, kde funkce **crypt()()** podporuje více typů šifrování se následující konstanty nastaví na 0 nebo 1 podle toho, jestli je daný typ dostupný:

- CRYPT_STD_DES - Standardní DES šifrování s dvouznakovým XXX SALT
- CRYPT_EXT_DES - Rozšířené DES šifrování s devítiznakovým XXX SALT
- CRYPT_MD5 - MD5 šifrování s dvanáctiznakovým XXX SALT začínajícím 1
- CRYPT_BLOWFISH - Rozšířené DES šifrování s šestnáctiznakovým XXX SALT začínajícím 2

Neexistuje žádná decrypt funkce, protože **crypt()** používá jednosměrný algoritmus.

Viz také: **md5()**.

echo (unknown)

Vytisknout jeden nebo více řetězců

```
echo (string arg1, string [argn]...)
```

Outputs all parameters.

Echo() vlastně není funkce (je to jazykový konstrukt), takže u něj nemusíte používat závorky.

Příklad 1. Ukázka echo()

```
echo "Hello World";

echo "Toto zabírá
několik řádků. Konce řádků se
vytisknou také";

echo "Toto zabírá\nněkolik řádků. Konce řádků se\nvytisknou také.";
```

Poznámka: Pokud chcete echo předat více argumentů, argumenty dokonce uzávorkovat nesmíte.

Viz také: **print()**, **printf()** a **flush()**.

explode (PHP 3, PHP 4)

Rozdělit řetězec na jiném řetězci

```
array explode (string separator, string string [, int limit])
```

Vrací pole řetězců, z nichž každý je částí argumentu *string* vzniklý jeho rozdělením na hranicích tvořených řetězcem *delim*. Pokud je definován *limit*, vrácené pole bude obsahovat maximálně *limit* prvků, a poslední prvek bude obsahovat celý zbytek *string*.

Poznámka: Argument *limit* byl přidán v PHP 4.0.1

Příklad 1. Ukázka explode()

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode (" ", $pizza);
```

Poznámka: I když **implode()** může z historických důvodů přijímat argumenty v obou možných pořadích, **explode()** nemůže. Musíte se ujistit, že je argument *separator* před argumentem *string*.

Viz také: **split()** a **implode()**.

get_html_translation_table (PHP 4 >= 4.0b4)

Vrátit překladovou tabulku používanou v **htmlspecialchars()** a **htmlentities()**

```
string get_html_translation_table  (int table [, int quote_style])
```

get_html_translation_table() vrací překladovou tabulku, která se interně používá ve funkcích **htmlspecialchars()** a **htmlentities()**. Dvě nové konstanty (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) vám umožňují určit, kterou tabulku chcete. A stejně jako u funkcí **htmlspecialchars()** a **htmlentities()** můžete případně určit *quote_style* se kterým pracujete. Defaultní hodnota je ENT_COMPAT mód. Viz popis těchto módů u **htmlspecialchars()**.

Příklad 1. Ukázka na překladovou tabulku

```
$trans = get_html_translation_table (HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr ($str, $trans);
```

Proměnná `$encoded` teď obsahuje: "Hallo & <Frau> & Krämer".

Skvělé je, že pomocí **array_flip()** můžete změnit směr překladu.

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Obsahem `$original` je: "Hallo & <Frau> & Krämer".

Poznámka: Tato funkce byla přidána v PHP 4.0.

Viz také: **htmlspecialchars()**, **htmlentities()**, **strtr()** a **array_flip()**.

get_meta_tags (PHP 3>= 3.0.4, PHP 4)

Získat hodnoty content atributů všech meta tagů v souboru a vrátit pole

```
array get_meta_tags (string filename [, int use_include_path])
```

Otevře *filename*, přečte ho po řádcích, a vyhledá <meta> tagy ve tvaru

Příklad 1. Ukázka na Meta Tagy

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!- tady čtení skončí ->
```

(věnujte pozornost koncům řádků - PHP používá ke čtení vstupu nativní funkci, takže Macovský soubor na Unixu nebude fungovat).

Hodnota atributu name se ve vráceném poli stává klíčem, hodnota atributu content hodnotou tohoto pole, takže ho můžete snadno projít nebo získat jednotlivé hodnoty pomocí snandardních funkcí pro práci s poli. Zvláštní znaky v hodnotě atributu name jsou nahrazeny znakem '_', zbytek se převede na malá písmena.

Pokud je *use_include_path* rovno 1, PHP se pokusí otevřít soubor v standardní include cestě.

hebrev (PHP 3, PHP 4)

Převést logický Hebrejský text na vizuální text

```
string hebrev (string hebrew_text [, int max_chars_per_line])
```

Volitelný argument *max_chars_per_line* indikuje maximální počet znaků na řádek výstupu. Funkce se snaží nerozdělovat slova.

Viz také: **hebrevc()**.

hebrevc (PHP 3, PHP 4)

Převést logický Hebrejský text na vizuální text s konverzí konců řádků

```
string hebrevc (string hebrew_text [, int max_chars_per_line])
```

Tato funkce se podobá **hebrev()** s tím rozdílem, že převádí konce řádků (\n) na "
\n". Volitelný argument *max_chars_per_line* indikuje maximální počet znaků na řádek výstupu. Funkce se snaží nerozdělovat slova.

Viz také: **hebrev()**.

htmlentities (PHP 3, PHP 4)

Převést všechny použitelné znaky na HTML entity

```
string htmlentities (string string [, int quote_style])
```

Tato funkce je ve všem shodná s **htmlspecialchars()** kromě toho, že na HTML entity se převedou všechny znaky, které mají odpovídající entity. Stejně jako **htmlspecialchars()** přijímá volitelný druhý argument, který indikuje, co se má stát s jednoduchými a dvojitými uvozovkami. ENT_COMPAT (default) převede pouze dvojité uvozovky, ENT_QUOTES převede dvojité i jednoduché uvozovky, a ENT_NOQUOTES ponechá jednoduché i dvojité uvozovky bez konverze.

V současnosti se používá znaková sada ISO-8859-1. Volitelný druhý argument byl přidán v PHP 3.0.17 a PHP 4.0.3.

Viz také: **htmlspecialchars()** a **nl2br()**.

htmlspecialchars (PHP 3, PHP 4)

Převést zvláštní znaky na HTML entity

```
string htmlspecialchars (string string [, int quote_style])
```

Některé znaky mají v HTML zvláštní význam, a pokud si mají zachovat běžný význam, měly by být reprezentovány HTML entitami. Tato funkce vrací řetězec, ve kterém došlo k některým z těchto konverzí; provádějí se ty překlady, které jsou v každodenním programování pro web nejužitečnější. Pokud požadujete překlad všech znakových entit HTML, použijte **htmlentities()**.

Tato funkce je užitečná, pokud se chcete chránit před případným výskytem HTML v textu dodaném uživateli, například u aplikací typu kniha hostů nebo diskusní skupina. Volitelný druhý argument, quote_style, určuje, co se má stát s jednoduchými a dvojitými uvozovkami. Defaultní mód, ENT_COMPAT, je zpětně kompatibilní mód, konvertuje pouze dvojité uvozovky a jednoduché uvozovky ponechává nepřeložené. Pokud zadáte ENT_QUOTES, přeloží se jednoduché i dvojité uvozovky, a pokud zadáte ENT_NOQUOTES, oba druhy zůstanou bez překladu.

Dochází k těmto překladům:

- ’&’ (ampersand) se stává ’&’
- ’"’ (dvojitá uvozovka) se stává ’"’ when ENT_NOQUOTES is not set.
- ”’ (jednoduchá uvozovka) se stává ’'’ only when ENT_QUOTES is set.
- ’<’ (menší než) se stává ’<’
- ’>’ (větší než) se stává ’>’

Příklad 1. Ukázka htmlspecialchars()

```
$new = htmlspecialchars("<a href=’test’>Test</a>", ENT_QUOTES);
```

Poznámka: tato funkce provádí pouze výše uvedené překlady. Kompletní překlad entit viz **htmlentities()**. Volitelný druhý argument byl přidán v PHP 3.0.17 a PHP 4.0.3.

Viz také: **htmlentities()** a **nl2br()**.

implode (PHP 3, PHP 4)

Spojit prvky pole pomocí řetězce

```
string implode (string glue, array pieces)
```

Vrací řetězec obsahující řetězcovou reprezentaci všech prvků pole v původním pořadí s řetězcem *glue* mezi každými dvěma prvky.

Příklad 1. Ukázka implode()

```
$colon_separated = implode (":", $array);
```

> **Poznámka: implode()** může z historických důvodů přijímat argumenty v obou možných pořadích. Kvůli konzistenci s **explode()** se ale doporučuje používat dokumentované pořadí argumentů.

Viz také: **explode()**, **join()** a **split()**.

join (PHP 3, PHP 4)

Spojit prvky pole pomocí řetězce

```
string join (string glue, array pieces)
```

Funkce **join()** je alias k **implode()**, a je ve všech ohledech stejná.

Viz také: **explode()**, **implode()** a **split()**.

levenshtein (PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

Spočítat XXX Levenshteinovu vzdálenost mezi dvěma řetězci

```
int levenshtein (string str1, string str2)
int levenshtein (string str1, string str2, int cost_ins, int cost_rep, int cost_del)
int levenshtein (string str1, string str2, function cost)
```

Tato funkce vrací XXX Levenshtein-Distance mezi předanými řetězci nebo -1, pokud délka jednoho z předaných řetězců přesáhne omezení 255 znaků (255 by mělo být pro běžná porovnání víc než dost, a nikdo se zdravým rozumem nebude v PHP dělat genetickou analýzu).

Levenshteinova vzdálenost se definuje jako minimální počet znaku, které musíte nahradit, vložit nebo smazat, abyste změnili *`str1`* na *`str2`*. Složitost tohoto algoritmu je `O(m*n)`, kde `n` a `m` jsou délky *`str1`* a *`str2`* (celkem slušné v porovnání se **similar_text()**, který je O(max(n,m)**3), ale i tak drahé).

Ve své nejjednodušší podobě tato funkce pouze vezme dva řetězce jako argumenty a spočítá počet vložení, nahrazení a smazání nutných k transformaci *`str1`* na *`str2`*.

Druhá varianta přijme tři další argumenty, které definují náklady na operace vložení, nahrazeni a smazání. Tato varianta je všeobecnější a přizpůsobivější než varianta první, ale ne tak výkonná.

Třetí varianta (zatím neimplementovaná) bude nejvšeobecnější a nejpřizpůsobivější, ale také nejpomalejší alternativou. Bude volat uživatelskou funkci, která určí náklady na všechny možné operace.

Tato uživatelská funkce se bude volat s následujícími argumenty:

- operatce, která se má provést: ’I’, ’R’ or ’D’
- původní znak v řetězci 1
- původní znak v řetězci 2
- pozice v řetězci 1
- pozice v řetězci 2
- znaky zbývající v řetězci 1
- znaky zbývající v řetězci 2

Tato uživatelská funkce musí vrátit kladné celé číslo vyčíslující náklady na tuto konkrétní operaci, ale může se rozhodnout použít pouze některé z přijatých argumentů.

Tento přístup nabízí možnost zohlednit důležitost určitých symbolů (znaků) a/nebo rozdíly mezi nimi, či dokonce kontext, ve kterém se vyskytují při určování nákladů na vložení, změnu nebo smazání, ale za cenu ztráty všech optimalizací využití CPU registru a XXX cache misses, které byly zapracovány do předchozích dvou variant.

Viz také: **soundex()**, **similar_text()** a **metaphone()**.

ltrim (PHP 3, PHP 4)

Odstranit netisknutelné znaky ze začátku řetězce

```
string ltrim (string str)
```

Tato funkce ořízne netisknutelné znaky ze začátku řetězce a vrací oříznutý řetězec. Netisknutelné znaky, které se v současnosti odstraňují, jsou: "\n", "\r", "\t", "\v", "\0", a prostá mezera.

Viz také: **chop()**, **rtrim()** a **trim()**.

md5 (PHP 3, PHP 4)

Spočítat MD5 XXX hash řetězce

```
string md5 (string str)
```

Spočítá MD5 XXX hash argumentu *str* pomocí MD5 Message-Digest Algoritmu společnosti RSA Data Security, Inc. (http://www.faqs.org/rfcs/rfc1321.html).

Viz také: **crc32()**.

Metaphone (PHP 4 >= 4.0b4)

Spočítat metaphone klíč řetězce

```
string metaphone (string str)
```

Spočítá metaphone klíč argumentu *str*.

metaphone(), podobně jako **soundex()**, vytvoří stejný klíč pro podobně znějící slova. Je přesnější než **soundex()**, protože zná základní pravidla anglické výslovnosti. Metaphone klíče mají proměnlivou délku.

Metaphone vyvinul Lawrence Philips <lphilips@verity.com>. Je popsáno v ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Poznámka: Tato funkce byla přidána v PHP 4.0.

nl2br (PHP 3, PHP 4)

Převede konce řádků na HTML konce řádků

```
string nl2br (string string)
```

Vrací *string*, ve kterém je před každý konec řádku vložen tag '
'.

Viz také: **htmlspecialchars()**, **htmlentities()** a **wordwrap()**.

Ord (PHP 3, PHP 4)

Vrátit ASCII hodnotu znaku

```
int ord (string string)
```

Vrací ASCII hodnotu prvního znaku v *`string`*. Tato funkce doplňuje **chr()**.

Příklad 1. Ukázka ord()

```
if (ord ($str) == 10) {
    echo "Prvním znakem \$str je line feed.\n";
}
```

Viz také: **chr()**.

parse_str (PHP 3, PHP 4)

Roparsovat řetězec do proměnných

```
void parse_str (string str [, array arr])
```

Rozparsuje řetězec jako kdyby to byl querystring předaný v URL a definuje příslušné proměnné v současném scope. Pokud je předán druhý argument *`arr`*, proměnné se místo toho uloží do této proměnné jako pole.

Příklad 1. Using parse_str()

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first;     /* vytiskne "value" */
echo $second[0]; /* vytiskne "this works" */
echo $second[1]; /* vytiskne "another" */
```

print (unknown)

Vytisknout řetězec

```
print (string arg)
```

Vytiskne *`arg`*.

Viz také: **echo()**, **printf()** a **flush()**.

printf (PHP 3, PHP 4)

Vytisknout formátovaný řetězec

```
int printf (string format [, mixed args...])
```

Vytvoří výtup podle argumentu *`format`*, který je popsán v dokumentaci **sprintf()**.

Viz také: **print()**, **sprintf()**, **sscanf()**, **fscanf()** a **flush()**.

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4)

Převést quoted-printable řetězec na osmibitový řetězec

```
string quoted_printable_decode  (string str)
```

Tato funkce vrací osmibitový binární řetězec odpovídající dekódovanému quoted printable řetězci. Tato funkce je podobná **imap_qprint()** s tou výjimkou, že tato funkce nevyžaduje IMAP modul.

quotemeta (PHP 3, PHP 4)

Opatřit lomítky metaznaky

```
string quotemeta (string str)
```

Vrací verzi *str* se zpětným lomítkem před všemi výskyty následujících znaků:

```
. \\ + * ? [ ^ ] ( $ )
```

Viz také: **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()** a **stripslashes()**.

rtrim (PHP 3, PHP 4)

Odstranit netisknutelné znaky z konce řetězce

```
string rtrim (string str)
```

Vrací předaný řetězec bez netisknutelných znaků (vč. konců řádku) na konci. Toto je alias k **chop()**.

Příklad 1. Ukázka rtrim()

```
$trimmed = rtrim ($line);
```

Viz také: **trim()**, **ltrim()** a **rtrim()**.

sscanf (PHP 4 >= 4.0.1)

Rozparsovat vstupní řetězec podle formátu

```
mixed sscanf (string str, string format [, string var1...])
```

Funkce **sscanf()** je vstupním analogem **printf()**. **sscanf()** čte řetězec *str* a interpretuje ho podle formátu *format*. Pokud jsou jí předány pouze dva argumenty, vrací rozparsované hodnoty v poli.

Příklad 1. Ukázka sscanf()

```
// zjištění sériového čísla
$serial = sscanf("SN/2350001","SN/%d");
// a data výroby
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "Zboží $serial bylo vyrobeno: $year-".substr($month,0,3)."-$day\n";
```

Pokud jsou jí předány volitelné argumenty, vrací tato funkce počet přiřazených hodnot. Volitelné argumenty musí být předány odkazem.

Příklad 2. Ukázka sscanf() - použití volitelných argumentů

```
// zjistit informace o autorovi a vygenerovat DocBook záznam
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
<firstname>$first</firstname>
<surname>$last</surname>
</author>\n";
```

Viz také: **fscanf()**, **printf()** a **sprintf()**.

setlocale (PHP 3, PHP 4)

Set locale information

```
string setlocale (string category, string locale)
```

category je řetězec určující kategorii funkcí ovlivněných nastavením locale:

- LC_ALL pro všechny níže uvedené kategorie
- LC_COLLATE pro porovnávání řetězců - v PHP v současnosti neimplementováno
- LC_CTYPE pro klasifikaci a konverzi znaků, např. **strtoupper()**
- LC_MONETARY pro localeconv() - v PHP v současnosti neimplementováno
- LC_NUMERIC pro oddělovač desetinných míst
- LC_TIME pro formátování data a času pomocí **strftime()**

Pokud je *locale* prázný řetězec (""), názvy locale se nastaví na hodnoty systémových proměnných se stejnými jmény jako mají výše uvedené kategorie, nebo z "LANG".

Pokud je *locale* nula nebo "0", locale se nezmění, pouze se vrátí současná hodnota.

setlocale() vrací nové aktuální locale nebo false, pokud na dotyčné platformě není funkcionalita locale implementována, zadané locale neexistuje, nebo je název kategorie neplatný. Neplatný název kategorie také vyvolá varování.

similar_text (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Spočítat podobnost dvou řetězců

```
int similar_text (string first, string second [, double percent])
```

similar_text() spočítá podobnost dvou řetězců podle Oliver [1993]. Pozn.: Tato implementace nepoužívá stack jako v Oliverově pseudokódu, nýbrž rekurzivní volání, což může či nemusí celý proces zrychlit. Komplexita tohoto algoritmu je O(N**3) kde N je délka nejdelšího řetězce.

Pokud je **similar_text()** předán třetí argument (odkazem), spočítá tato funkce podobnost v procentech. Vrací počet znaků shodných v obou řetězcích.

soundex (PHP 3, PHP 4)

Spočítat soundex klíč řetězce

```
string soundex (string str)
```

Spočítá soundex klíč *str*.

Soundex klíče mají tu vlastnost, že slova vyslovovaná podobně produkují shodné soundex klíče, a dají se proto využít ke zjednodušení hledání v databázích, kde znáte výslovnost, ale ne hláskování. Tato funkce vrací řetězec dlouhý 4 znaky začínající písmenem.

Tato konkrétní soundex funkce je popsána Donaldem Knuthem v "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Příklad 1. Soundex ukázky

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Hilbert") == soundex ("Heilbronn") == 'H416';
soundex ("Knuth") == soundex ("Kant") == 'K530';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf (PHP 3, PHP 4)

Vrátit formátovaný řetězec

```
string sprintf (string format [, mixed args...])
```

Vrací řetězec vytvořený podle formátovacího řetězce *format*.

Formátovací řetězec se skládá z nula nebo více direktiv: běžných znaků (kromě %), které se přímo kopírují do výsledku, a *převodních specifikací*, z nichž každá přijímá jeden argument. Toto platí pro **sprintf()** i **printf()**.

Každá převodní specifikace se skládá ze znaku procenta (%), následovaného jedním nebo více z těchto znaků, v tomto pořadí:

1. Volitelný *padding specifier*, který určuje, jaký znak se použije na doplnění výsledku na správnou délku řetězce. Může to být mezera nebo `0` (písmeno nula). Default je nula. Jiný doplňující znak můžete zadat tak, že před něj předřadíte jednoduchou uvozovku (`'`). Viz ukázky níže.
2. Volitelný *alignment specifier*, který určuje, jestli se má výsledek zarovnat doleva nebo doprava. Default je doprava, pomlčka (`-`) to změní na doleva.
3. Volitelné číslo *width specifier*, které určuje, kolik znaků (minimálně) má obsahovat výsledek převodu.
4. Volitelný *precision specifier*, který určuje, kolik desetinných míst se má zobrazit u čísel s desetinnou čárkou. Tento přepínač nemá žádný vliv na jiné typy než double. (Další funkcí užitečnou na formátování čísel je **number_format()**.)
5. *type specifier*, který určuje, za jaký typ se mají data argumentu považovat. Možné typy:

   ```
   % - a doslovný znak procenta. Nevyžaduje se žádný argument.
   b - argument se považuje za integer a je prezentován jako binární číslo.
   c - argument se považuje za integer a je prezentován jako znak s touto ASCII hodnotou.
   d - argument se považuje za integer a je prezentován jako desítkové číslo.
   f - argument se považuje za double a je prezentován jako číslo s plovoucí desetinou čárkou.
   o - argument se považuje za integer a je prezentován jako oktalové číslo.
   s - argument se považuje za řetězec a je takto prezentován.
   x - the argument se považuje za integer a je prezentován jako hexadecimální číslo (s malými
   X - argument se považuje za integer a je prezentován jako hexadecimální číslo (s kapitálkam
   ```

Viz také: **printf()**, **sscanf()**, **fscanf()** a **number_format()**.

Příklad 1. Ukázka sprintf(): zero-padded integers

```
$isodate = sprintf ("%04d-%02d-%02d", $year, $month, $day);
```

Příklad 2. Ukázka sprintf(): formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf ("%01.2f", $money);
// echo $formatted will output "123.10"
```

strncasecmp (PHP 4 >= 4.0.2)

Binárně bezpečné case-insensitive porovnání prvních n znaků řetězců

```
int strncasecmp (string str1, string str2, int len)
```

Tato funkce se podobá **strcasecmp()**, s tím rozdílem, že můžete určit (maximální) počet znaků (*len*) z každého z řetězců, které se použijí při porovnání. Pokud je některý z řetězců kratší než *len*, pak se pro porovnání použije délka tohoto řetězce.

Pokud je *str1* méně než *str2*, vrací < 0; pokud je *str1* větší než *str2*, vrací > 0, a 0, pokud jsou stejné.

Viz také: **ereg()**, **strcasecmp()**, **strcmp()**, **substr()**, **stristr()** a **strstr()**.

strcasecmp (PHP 3>= 3.0.2, PHP 4)

Binárně bezpečné case-insensitive porovnání řetězců

```
int strcasecmp (string str1, string str2)
```

Pokud je *str1* méně než *str2* vrací < 0; pokud je *str1* větší než *str2* vrací > 0, a 0, pokud jsou stejné.

Příklad 1. Ukázka strcasecmp()

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp ($var1, $var2)) {
    echo ’v case-insensitive textovém porovnání se $var1 rovná $var2’;
}
```

Viz také: **ereg()**, **strcmp()**, **substr()**, **stristr()**, **strncasecmp()** a **strstr()**.

strchr (PHP 3, PHP 4)

Najít první výskyt znaku

```
string strchr (string haystack, string needle)
```

Tato funkce je alias k **strstr()**, a je ve všech směrech identická.

strcmp (PHP 3, PHP 4)

Binárně bezpečně porovnat řetězce

```
int strcmp (string str1, string str2)
```

Pokud je *str1* méně než *str2*, vrací < 0; pokud je *str1* větší než *str2*, vrací > 0, a 0, pokud jsou stejné.

Pozn.: toto srovnání je case-sensitive.

Viz také: **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strncasecmp()**, **strncmp()** a **strstr()**.

strcspn (PHP 3>= 3.0.3, PHP 4)

Najít délku úvodního segmentu neodpovídajícího masce

```
int strcspn (string str1, string str2)
```

Vrací délku úvodního segmentu *str1*, který *ne*obsahuje žádný ze znaků *str2*.

Viz také: **strspn()**.

strip_tags (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Odstranit z řetězce HTML a PHP tagy

```
string strip_tags (string str [, string allowable_tags])
```

Tato funkce se snaží odstranit z předaného řetězce všechny HTML a PHP tagy. It errors on the side of caution in case of incomplete or bogus tags. It uses the same tag stripping state machine as the **fgetss()** function.

Volitelný druhý argument můžete použít k určení tagů, které se nemají odstranit.

Poznámka: Argument *allowable_tags* byl přidán v PHP 3.0.13, PHP 4 b3.

stripcslashes (PHP 4 >= 4.0b4)

Un-quote string quoted with **addcslashes()**

```
string stripcslashes (string str)
```

Vrací řetězec bez odstraněných zpětných lomítek. Rozeznává Céčkové \n, \r ..., oktalové a hexadecimální reprezentace.

Poznámka: Tato funkce byla přidána v PHP4b3-dev.

Viz také: **addcslashes()**.

stripslashes (PHP 3, PHP 4)

Un-quote string quoted with **addslashes()**

```
string stripslashes (string str)
```

Vrací řetězec bez odstraněných zpětných lomítek. (`\'` se stává `'` a pod.) Zdvojená zpětná lomítka se spojují do jednoduchých.

Viz také: **addslashes()**.

stristr (PHP 3>= 3.0.6, PHP 4)

Case-insensitive **strstr()**

```
string stristr (string haystack, string needle)
```

Vrací *`haystack`* od prvního výskytu *`needle`* do konce. *`needle`* a *`haystack`* se zkoumají bez ohledu na velikost písmen.

Pokud *`needle`* nenajde, vrací `false`.

Pokud *`needle`* není řetězec, převede se na integer a použije se jako XXX ordinal hodnota znaku.

Viz také: **strchr()**, **strrchr()**, **substr()** a **ereg()**.

strlen (PHP 3, PHP 4)

Zjistit délku řetězce

```
int strlen (string str)
```

Vrací délku (počet znaků) argumentu *`string`*.

strnatcmp (PHP 4 >= 4.0RC2)

Porovnání řetězců algoritmem "přirozeného třídění"

```
int strnatcmp (string str1, string str2)
```

Tato funkce implementuje srovnávací algoritmus který třídí alfanumerické řetězce stejným způsobem jako člověk, toto se popisuje jako "přirozené třídění". Ukázka rozdílu mězi tímto algoritmem a běžnými počítačovými algoritmy pro řazení řetězců (např. **strcmp()**):

```
$arr1 = $arr2 = array ("img12.png","img10.png","img2.png","img1.png");
echo "Standardní porovnávání řetězců\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nPřirozené porovnávání řetězců\n";
usort($arr2,"strnatcmp");
print_r($arr2);
```

Výše uvedený kód vygeneruje následující výstup:

```
Standardní porovnávání řetězců
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Přirozené porovnávání řetězců
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Více informací viz stránka Martina Poola Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Podobně jako jiné funkce pro porovnávání řetězců i tato vrací < 0 pokud je *`str1`* menší než *`str2`*; > 0 pokud je *`str1`* větší než *`str2`*, a 0 pokud jsou shodné.

Pozn.: toto porovnání je case-sensitive.

Viz také: **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strncasecmp()**, **strnatcasecmp()**, **strstr()**, **natsort()** a **natcasesort()**.

strnatcasecmp (PHP 4 >= 4.0RC2)

Case-insensitive textové porovnání s využitím "natural order" algoritmu

```
int strnatcasecmp (string str1, string str2)
```

Tato funkce implementuje srovnávací algoritmus který třídí alfanumerické řetězce stejným způsobem jako člověk. Chování této funkce se podobá **strnatcmp()** s tou výjimkou, že porovnání je case-insensitive. Více informací viz stránka Martina Poola Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Podobně jako jiné funkce pro porovnávání řetězců i tato vrací < 0 pokud je *`str1`* menší než *`str2`*; > 0 pokud je *`str1`* větší než *`str2`*, a 0 pokud jsou shodné.

Viz také: **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strncasecmp()**, **strnatcmp()** a **strstr()**.

strncmp (PHP 4 >= 4.0b4)

Binárně bezpečné porovnání prvních n znaků v řetězcích

```
int strncmp (string str1, string str2, int len)
```

This function is similar to **strcmp()**, with the difference that you can specify the (upper limit of the) number of characters (*`len`*) from each string to be used in the comparison. If any of the strings is shorter than *`len`*, then the length of that string will be used for the comparison.

Vrací < 0 pokud je *`str1`* menší než *`str2`*; > 0 pokud je *`str1`* větší než *`str2`*, a 0 pokud jsou shodné.

Pozn.: toto srovnání je case-sensitive.

Viz také: **ereg()**, **strncasecmp()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()** a **strstr()**.

str_pad (PHP 4 >= 4.0.1)

Doplnit řetězec jiným řetězcem na určitou délku

```
string str_pad (string input, int pad_length [, string pad_string [, int pad_type]])
```

str_pad() doplní řetězec *input* zleva, zprava nebo z obou stran na danou délku. Pokud jí není předán volitelný argument *pad_string*, doplní se *input* mezerami, jinak se doplní znaky z *pad_string*.

Volitelný argument *pad_type* může nabýt hodnot STR_PAD_RIGHT, STR_PAD_LEFT nebo STR_PAD_BOTH. Default je STR_PAD_RIGHT.

Pokud je hodnota *pad_length* negativní nebo menší než je délka *input*, k doplnění nedojde.

Příklad 1. Ukázka str_pad()

```
$input = "Alien";
print str_pad($input, 10);                      // produces "Alien     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Alien"
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Alien___"
```

strpos (PHP 3, PHP 4)

Najít pozici prvního výskytu řetězce

```
int strpos (string haystack, string needle [, int offset])
```

Vrací číselnou pozici prvního výskytu *needle* v řetězci *haystack*. Narozdíl od **strrpos()** tato funkce přijme jako argument *needle* řetězec více znaků, a celý tento řetězec se použije.

Pokud *needle* nenajde, vrací `false`.

Poznámka: Návratové hodnoty "znak nalezen na pozici 0" a "znak nenalezen" se dají snadno zaměnit. Tady je návod, jak zjistit tento rozdíl:

```
// v PHP 4.0b3 a novějších:
$pos = strpos ($mystring, "b");
if ($pos === false) { // tři rovnítka
    // nenalezeno...
}

// ve verzích starších než 4.0b3:
$pos = strpos ($mystring, "b");
if (is_string ($pos) && !$pos) {
    // nenalezeno...
}
```

Pokud *needle* není řetězec, převede se na integer a použije se jako XXX ordinal hodnota znaku.

Volitelný argument *offset* vám umožňuje určit na které pozici v *haystack* má hledání začít. Vrácená pozice je i tak relativní k začátku *haystack*.

Viz také: **strrpos()**, **strrchr()**, **substr()**, **stristr()** a **strstr()**.

strrchr (PHP 3, PHP 4)

Najít poslední výskyt znaku v řetězci

```
string strrchr (string haystack, string needle)
```

Tato funkce vrací tu část *haystack*, která začíná poslením výskytem *needle* a pokračuje do konce *haystack*.

Pokud *needle* nenajde, vrací false.

Pokud *needle* obsahuje více než jeden znak, použije se první z nich.

Pokud *needle* není řetězec, převede se na integer a použije se jako XXX ordinal hodnota znaku.

Příklad 1. Ukázka strrchr()

```
// získat poslední adresář v $PATH
$dir = substr (strrchr ($PATH, ":"), 1);

// získat všechno po posledním konci řádku
$text = "Řádek 1\nŘádek 2\nŘádek 3";
$last = substr (strrchr ($text, 10), 1 );
```

Viz také: **substr()**, **stristr()** a **strstr()**.

str_repeat (PHP 4 >= 4.0b4)

Opakovat řetězec

```
string str_repeat (string input, int multiplier)
```

Vrací *input_str* *multiplier* krát opakovaný. *multiplier* musí být větší než 0.

Příklad 1. Ukázka str_repeat()

```
echo str_repeat ("-=", 10);
```

This will output "-=-=-=-=-=-=-=-=-=-=".

Poznámka: Tato funkce byla přidána v PHP 4.0.

strrev (PHP 3, PHP 4)

Obrátit řetězec

```
string strrev (string string)
```

Vrací *string* v opačném pořadí.

strrpos (PHP 3, PHP 4)

Najít pozici posledního výskytu znaku v řetězci

```
int strrpos (string haystack, char needle)
```

Vrací číselnou pozici posledního výskytu *needle* v řetězci *haystack*. *needle* může být jen jeden znak dlouhá. Pokud obsahuje více znaků, použije se první z nich.

Pokud *needle* nenajde, vrací false.

Poznámka: Návratové hodnoty "znak nalezen na pozici 0" a "znak nenalezen" se dají snadno zaměnit. Tady je návod, jak zjistit tento rozdíl:

```
// v PHP 4.0b3 a novějších:
$pos = strpos ($mystring, "b");
if ($pos === false) { // tři rovnítka
    // nenalezeno...
}

// ve verzích starších než 4.0b3:
$pos = strpos ($mystring, "b");
if (is_string ($pos) && !$pos) {
    // nenalezeno...
}
```

Pokud *needle* není řetězec, převede se na integer a použije se jako XXX ordinal hodnota znaku.

Viz také: **strpos()**, **strrchr()**, **substr()**, **stristr()** a **strstr()**.

strspn (PHP 3>= 3.0.3, PHP 4)

Zjistit délku úvodního segmentu odpovídajícího masce

```
int strspn (string str1, string str2)
```

Vrací úvodního segmentu *str1*, který se skládá výhradně ze znaků v *str2*.

```
strspn ("42 je odpověd', co je otázka...", "1234567890");
```

vrátí 2.

Viz také: **strcspn()**.

strstr (PHP 3, PHP 4)

Najít první výskyt řetězce

```
string strstr (string haystack, string needle)
```

Vrací část *haystack* od prvního výskytu *needle* do konce.

Pokud *needle* nenajde, vrací false.

Pokud *needle* není řetězec, převede se na integer a použije se jako XXX ordinal hodnota znaku.

Poznámka: Pozn.: tato funkce je case-sensitive. Pro case-insensitive hledání použijte **stristr()**.

Příklad 1. Ukázka strstr()

```
$email = 'sterling@designmultimedia.com';
$domain = strstr ($email, '@');
print $domain; // vytiskne @designmultimedia.com
```

Viz také: **stristr()**, **strrchr()**, **substr()** a **ereg()**.

strtok (PHP 3, PHP 4)

Tokenize string

```
string strtok (string arg1, string arg2)
```

strtok() is used to tokenize a string. That is, if you have a string like "This is an example string" you could tokenize this string into its individual words by using the space character as the token.

Příklad 1. Ukázka strtok()

```
$string = "This is an example string";
$tok = strtok ($string," ");
while ($tok) {
    echo "Word=$tok<br>";
    $tok = strtok (" ");
}
```

Note that only the first call to strtok uses the string argument. Every subsequent call to strtok only needs the token to use, as it keeps track of where it is in the current string. To start over, or to tokenize a new string you simply call strtok with the string argument again to initialize it. Note that you may put multiple tokens in the token parameter. The string will be tokenized when any one of the characters in the argument are found.

Also be careful that your tokens may be equal to "0". This evaluates to false in conditional expressions.

Viz také: **split()** a **explode()**.

strtolower (PHP 3, PHP 4)

Změnit řetězec na malá písmena

```
string strtolower (string str)
```

Vrací *`string`* se všemi alfabetickými znaky změněnými na malá písmena.

Pozn.: co je 'alfabetický' je dáno aktuálním místním nastavením. Například ve standardním "C" locale se znaky jako přehlasované a (Ä) nepřevedou.

Příklad 1. Ukázka strtolower()

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
```

```
print $str; # vytiskne mary had a little lamb and she loved it so
```

Viz také: **strtoupper()** a **ucfirst()**.

strtoupper (PHP 3, PHP 4)

Změnit řetězec na velká písmena

```
string strtoupper (string string)
```

Vrací *string* se všemi alfabetickými znaky změněnými na velká písmena.

Pozn.: co je 'alfabetický' je dáno aktuálním místním nastavením. Například ve standardním "C" locale se znaky jako přehlasované a (ä) nepřevedou.

Příklad 1. Ukázka strtoupper()

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper ($str);
print $str; # vytiskne MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

Viz také: **strtolower()** a **ucfirst()**.

str_replace (PHP 3>= 3.0.6, PHP 4)

Nahradit všechny výskyty jednoho řetězce v jiném dalším řetězcem

```
string str_replace (string needle, string str, string haystack)
```

Tato funkce nahradí všechny výskyty *needle* v argumentu *haystack* argumentem *str*. Pokud nepotřebujete složitá pravidla pro nahrazování, měli byste vždy použít tuto funkci místo **ereg_replace()**.

Příklad 1. Ukázka str_replace()

```
$bodytag = str_replace ("%body%", "black", "<body text=%body%>");
```

Tato funkce je XXX binárně bezpečná.

> **Poznámka:** Funkce **str_replace()** byla přidána v PHP 3.0.6, ale až do verze PHP 3.0.8 fungovala špatně.

Viz také: **ereg_replace()** a **strtr()**.

strtr (PHP 3, PHP 4)

Přeložit určité znaky

```
string strtr (string str, string from, string to)
```

Tato funkce upraví *str* tak, že všechny výskyty všech znaků ve *from* přeloží na odpovídající znaky v *to* a vrátí výsledek.

Pokud jsou *from* a *to* různě dlouhé, přebývající znaky z delšího z těch dvou se ignorují.

Příklad 1. Ukázka strtr()

```
$addr = strtr($addr, "äĺö", "aao");
```

strtr() se dá také volat pouze se dvěma argumenty. Při volání se dvěma argumenty se chová takto: *from* musí být pole obsahující páry řetězců, které se zamění ve zdrojovém řetězci. **strtr()** vždy hledá nejdelší možnou shodu a *NENAHRAZUJE* ty části řetězce, na kterých už pracovala.

Ukázky:

```
$trans = array ("ahoj" => "nazdar", "nazdar" => "ahoj");
echo strtr("nazdar lidi, řekl jsem ahoj", $trans) . "\n";
```

Výsledek: "ahoj lidi, řekl jsem nazdar",

> **Poznámka:** Tato vlastnost (dva argumenty) byla přidána v PHP 4.0.

Viz také: **ereg_replace()**.

substr (PHP 3, PHP 4)

Vrátit část řetězce

```
string substr (string string, int start [, int length])
```

substr() vrací část argumentu *string* určenou argumenty *start* a *length*.

Pokud je *start* pozitivní, vrácený řetězec začne *start*-tým znakem řetězce *string*, počítáno od nuly. Například v řetězci 'abcdef' je znakem na 0-té pozici 'a', znakem na pozici 2 je 'c', atd.

Příklady:

```
$rest = substr ("abcdef", 1);    // vrátí "bcdef"
$rest = substr ("abcdef", 1, 3); // vrátí "bcd"
```

Pokud je *start* negativní, vrácený řetězec začne *start*-tým znakem od konce argumentu *string*.

Příklady:

```
$rest = substr ("abcdef", -1);    // vrátí "f"
$rest = substr ("abcdef", -2);    // vrátí "ef"
$rest = substr ("abcdef", -3, 1); // vrátí "d"
```

Pokud je argument *length* kladný, vrácený řetězec skončí *length* znaků od *start*. Pokud by to znamenalo řetězec se zápornou délkou (*start* je za *length*), vrácený řetězec bude sestávat z jediného znaku na pozici *start*.

Pokud je argument *length* kladný, vrácený řetězec skončí *length* znaků od konce argumentu *string*. Pokud by to znamenalo řetězec se zápornou délkou, vrácený řetězec bude sestávat z jediného znaku na pozici *start*.

Příklady:

```
$rest = substr ("abcdef", 1, -1); // vrátí "bcde"
```

Viz také: **strrchr()** a **ereg()**.

substr_count (PHP 4 >= 4.0RC2)

Spočítat počet výskytů řetězce

```
int substr_count (string haystrack, string needle)
```

substr_count() vrací počet výskytů řetězce *needle* v řetězci *haystack* string.

Příklad 1. Ukázka substr_count()

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace (PHP 4 >= 4.0b4)

Nahradit část řetězce jiným řetězcem

```
string substr_replace (string string, string replacement, int start [, int length])
```

substr_replace() nahrazuje část řetězce *string* ohraničenou argumenty *start* a (volitelně) *length* řetězcem v argumentu *replacement*. Vrací výsledek.

Pokud je *start* pozitivní, náhrada začne na *start*-tém znaku argumentu *string*.

Pokud je *start* negativní, náhrada začne na *start*-tém znaku od konce argumentu *string*.

Pokud je přítomen *length*, a je pozitivní, představuje délku části argumentu *string*, která bude nahražena. Pokud je negativní, představuje počet znaků od konce *string*, kde má nahrazování skončit. Pokud přítomen není, bere se standardně strlen(*string*); tj. nahrazování končí na konci argumentu *string*.

Příklad 1. Ukázka substr_replace()

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Originál: $var<hr>\n";

/* Tyto dva příklady nahradí celý obsah proměnné $var řetězcem 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Toto vloží 'bob' na začátek $var. */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Tyto dva příklady nahradí 'MNRPQR' ve $var řetězcem 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Toto z $var odstraní 'MNRPQR'. */
echo substr_replace ($var, '', 10, -1) . "<br>\n";
?>
```

Viz také **str_replace()** a **substr()**.

Poznámka: Funkce **substr_replace()** byla přidána v PHP 4.0.

trim (PHP 3, PHP 4)

Odstranit netisknutelné znaky ze začátku a konce řetězce

```
string trim (string str)
```

Tato funkce odstraňuje netisknutelné znaky ze začátku a konce řetězce a vrací řetězec bez těchto znaků. Netisknutelné znaky, které se v současnosti odstraňují, jsou: "\n", "\r", "\t", "\v", "\0", a mezera.

Viz také **chop()**, **rtrim()** a **ltrim()**.

ucfirst (PHP 3, PHP 4)

Změní první písmeno řetězce na velké

```
string ucfirst (string str)
```

Změní první znak argumentu *str*, pokud je tento znak alfabetický.

Pozn.: co znamená 'alfabetický' určuje aktuální místní nastavení (locale). Například ve standardním "C" locale se znaky jako přehlasované a (ä) nepřevedou.

Příklad 1. Ukázka ucfirst()

```
$text = 'mary had a little lamb and she loved it so.';
$text = ucfirst ($text); // $text is now Mary had a little lamb
                         // and she loved it so.
```

Viz také **strtoupper()** a **strtolower()**.

ucwords (PHP 3>= 3.0.3, PHP 4)

Změnit první znak každého slova v řetězci na velké písmeno

```
string ucwords (string str)
```

Změní první znak každého slova v argumentu *str* na velké písmeno, pokud je tento znak alfabetický.

Příklad 1. Ukázka ucwords()

```
$text = "mary had a little lamb and she loved it so.";
$text = ucwords($text); // $text is now: Mary Had A Little
                        // Lamb And She Loved It So.
```

> **Poznámka:** Definice slova je: jakýkoli řetězec znaků, který následuje bezprostředně po netisknutelném znaku (to jsou: mezera, posun o tiskouvou stranu, přesun na novou řádku, návrat vozíku, horizontální tabelátor a vertikální tabelátor0.

Viz také **strtoupper()**, **strtolower()** and **ucfirst()**.

wordwrap (PHP 4 >= 4.0.2)

Zalámat řetězec na daný počet znaků pomocí break znaku

```
string wordwrap (string str [, int width [, string break [, int cut]]])
```

Zaláme řetězec *str* na sloupci určeném (volitelným) argumentem *break*.

Pokud není zadán argument *width* nebo *break*, **wordwrap()** automaticky zaláme řádky řetězce na sloupci 75 znakem '\n' (konec řádku).

Pokud má argument *cut* hodnotu 1, řetězec se na určenou šířku zalomí vždy. Takže pokud máte slovo delší než je daná šířka, rozdělí se. (Viz druhý příklad.)

Poznámka: Argument *cut* byl přidán PHP 4.0.3.

Příklad 1. Ukázka wordwrap()

```
$text = "Rychlá hnědá liška přeskočila líného psa.";
$newtext = wordwrap( $text, 20 );

echo "$newtext\n";
```

Tato ukázka by zobrazila:

```
Rychlá hnědá liška
přeskočila líného psa.
```

Příklad 2. Ukázka wordwrap()

```
$text = "Velmi dlouhé slooooooooooovo.";
$newtext = wordwrap( $text, 8, "\n", 1);

echo "$newtext\n";
```

Tato ukázka by zobrazila:

```
Velmi
dlouhé
slooooo
ooooovo.
```

Viz také: **nl2br()**.

LXXVI. Sybase functions

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4)

get number of affected rows in last query

```
int sybase_affected_rows ([int link_identifier])
```

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

> This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use **sybase_num_rows()**.
>
> **Poznámka:** This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close (PHP 3, PHP 4)

close Sybase connection

```
bool sybase_close (int link_identifier)
```

Returns: true on success, false on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by sybase_pconnect().

See also: **sybase_connect()**, **sybase_pconnect()**.

sybase_connect (PHP 3, PHP 4)

open Sybase server connection

```
int sybase_connect (string servername, string username, string password [, string
charset])
```

Returns: A positive Sybase link identifier on success, or false on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to sybase_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **sybase_close()**.

See also **sybase_pconnect()**, **sybase_close()**.

sybase_data_seek (PHP 3, PHP 4)

move internal row pointer

```
bool sybase_data_seek (int result_identifier, int row_number)
```

Returns: true on success, false on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specifyed row number. The next call to **sybase_fetch_row()** would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4)

fetch row as array

```
array sybase_fetch_array (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

sybase_fetch_array() is an extended version of **sybase_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using sybase_fetch_array() is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

For further details, also see **sybase_fetch_row()**

sybase_fetch_field (PHP 3, PHP 4)

get field information

```
object sybase_fetch_field (int result [, int field_offset])
```

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by sybase_fetch_field() is retreived.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See also **sybase_field_seek()**

sybase_fetch_object (PHP 3, PHP 4)

fetch row as object

```
int sybase_fetch_object (int result)
```

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

sybase_fetch_object() is similar to **sybase_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **sybase_fetch_array()**, and almost as quick as **sybase_fetch_row()** (the difference is insignificant).

See also: **sybase_fetch-array()** and **sybase_fetch-row()**.

sybase_fetch_row (PHP 3, PHP 4)

get row as enumerated array

```
array sybase_fetch_row (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to sybase_fetch_rows() would return the next row in the result set, or false if there are no more rows.

See also: **sybase_fetch_array()**, **sybase_fetch_object()**, **sybase_data_seek()**, **sybase_fetch_lengths()**, and **sybase_result()**.

sybase_field_seek (PHP 3, PHP 4)

set field offset

```
int sybase_field_seek (int result, int field_offset)
```

Seeks to the specified field offset. If the next call to **sybase_fetch_field()** won't include a field offset, this field would be returned.

See also: **sybase_fetch_field()**.

sybase_free_result (PHP 3, PHP 4)

free result memory

```
bool sybase_free_result (int result)
```

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_get_last_message (PHP 3, PHP 4)

Returns the last message from the server

```
string sybase_get_last_message (void )
```

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity (PHP 3, PHP 4)

Sets minimum client severity

```
void sybase_min_client_severity (int severity)
```

sybase_min_client_severity() sets the minimum client severity level.

Poznámka: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: **sybase_min_server_severity()**.

sybase_min_error_severity (PHP 3, PHP 4)

Sets minimum error severity

```
void sybase_min_error_severity (int severity)
```

sybase_min_error_severity() sets the minimum error severity level.
See also: **sybase_min_message_severity()**.

sybase_min_message_severity (PHP 3, PHP 4)

Sets minimum message severity

```
void sybase_min_message_severity (int severity)
```

sybase_min_message_severity() sets the minimum message severity level.
See also: **sybase_min_error_severity()**.

sybase_min_server_severity (PHP 3, PHP 4)

Sets minimum server severity

```
void sybase_min_server_severity (int severity)
```

sybase_min_server_severity() sets the minimum server severity level.

Poznámka: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: **sybase_min_client_severity()**.

sybase_num_fields (PHP 3, PHP 4)

get number of fields in result

```
int sybase_num_fields (int result)
```

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, **sybase_query()**, **sybase_fetch_field()**, **sybase_num_rows()**.

sybase_num_rows (PHP 3, PHP 4)

get number of rows in result

```
int sybase_num_rows (int result)
```

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, **sybase_query()** and, **sybase_fetch_row()**.

sybase_pconnect (PHP 3, PHP 4)

open persistent Sybase connection

```
int sybase_pconnect (string servername, string username, string password [, string
charset])
```

Returns: A positive Sybase persistent link identifier on success, or false on error

sybase_pconnect() acts very much like **sybase_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**sybase_close()** will not close links established by **sybase_pconnect()()**).

This type of links is therefore called 'persistent'.

sybase_query (PHP 3, PHP 4)

send Sybase query

```
int sybase_query (string query, int link_identifier)
```

Returns: A positive Sybase result identifier on success, or false on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **sybase_connect()** was called, and use it.

See also: **sybase_db_query()**, **sybase_select_db()**, and **sybase_connect()**.

sybase_result (PHP 3, PHP 4)

get result data

```
string sybase_result (int result, int row, mixed field)
```

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **sybase_fetch_row()**, **sybase_fetch_array()**, and **sybase_fetch_object()**.

sybase_select_db (PHP 3, PHP 4)

select Sybase database

```
bool sybase_select_db (string database_name, int link_identifier)
```

Returns: true on success, false on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **sybase_connect()** was called, and use it.

Every subsequent call to **sybase_query()** will be made on the active database.

See also: **sybase_connect()**, **sybase_pconnect()**, and **sybase_query()**

LXXVII. URL Functions

base64_decode (PHP 3, PHP 4)

Dekódovat data kódovaná pomocí MIME base64

```
string base64_decode (string encoded_data)
```

Base64_decode() dekóduje *encoded_data* a vrátí původní data. Vrácená data mohou být binární

Viz také: **base64_encode()**, RFC 2045 sekce 6.8.

base64_encode (PHP 3, PHP 4)

Zakódovat data pomocí MIME base64

```
string base64_encode (string data)
```

Base64_encode() vrátí *data* zakódovaný pomocí base64. Toto kódování je navrženo tak, aby umožnilo binárním datům přežít transport přenosovými vrstvami, které nejsou osmibitové, jako jsou například těla emailů.

Data kódovaná pomocí base64 zabírají o zhruba 33% prostoru více než původní data.

Viz také: **base64_decode()**, **chunk_split()**, RFC-2045 sekce 6.8.

parse_url (PHP 3, PHP 4)

Rozebrat URL a vrátit její komponenty

```
array parse_url (string url)
```

Tato funkce vrátí asociativní pole všech komponent URL přítomnych v *url*. Ty mohou být: "scheme", "host", "port", "user", "pass", "path", "query" a "fragment".

rawurldecode (PHP 3, PHP 4)

Dekódovat URL-kódovaný řetězec

```
string rawurldecode (string str)
```

Vrátí řetězec, ve kterém sekvence znaku procent (%) následových dvěma šestnáctkovými číslicemi byly nahrazeny prostými znaky. Například řetězec

```
foo%20bar%40baz
```

dekóduje na

```
foo bar@baz
```

.

Viz také: **rawurlencode()**, **urldecode()**, **urlencode()**.

rawurlencode (PHP 3, PHP 4)

URL-kódovat podle RFC1738

```
string rawurlencode (string str)
```

Vrátí řetězec, ve kterém byly všechny nealfanumerické znaky kromě

```
-_.
```

nahrazeny znakem procent (%) následovaným dvěma šestnáctkovými číslicemi. To je kódování popsané v RFC1738 na ochranu prostých znaků před interpretací jako zvláštní oddělovače v URL a na ochranu URL před komolením přenosovými systémy se znakovými konvencemi (jako jsou některé emailové systémy). Například, pokud chcete k FTP URL přidat heslo:

Příklad 1. Příklad rawurlencode() č. 1

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Nebo, pokud předáváte informace v komponentě URL obsahující info o cestě:

Příklad 2. Příklad rawurlencode() č. 2

```
echo '<A HREF="http://x.com/department_list_script/',
    rawurlencode ('sales and marketing/Miami'), '">';
```

Viz také: **rawurldecode()**, **urldecode()**, **urlencode()**.

urldecode (PHP 3, PHP 4)

Dekódovat URL-kódovaný řetězec

```
string urldecode (string str)
```

Dekóduje všechny %## kódy v daném řetězci. Vrátí dekódovaný řetězec.

Příklad 1. Příklad urldecode()

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
    $b = split ('=', $a [$i]);
    echo 'Hodnota argumentu ', htmlspecialchars (urldecode ($b [0])),
         ' je ', htmlspecialchars (urldecode ($b [1])), "<BR>";
    $i++;
}
```

Viz také **urlencode()**, **rawurlencode()**, **rawurldecode()**.

urlencode (PHP 3, PHP 4)

URL-kódovat řetězec

```
string urlencode (string str)
```

Vrátí řetězec, ve kterém byly všechny nealfanumerické znaky kromě -_. nahrazeny znakem procent (%) následovaným dvěma šestnácktovými číslicemi a mezery kódovány jako znaky plus (+). Kódování je stejné jako u dat postovaných z

WWW formuláře, tj. stejně jako u `application/x-www-form-urlencoded` typu. To se liší od RFC1738 kódování (viz **rawurlencode()**) v tom, že z historických důvodů se mezery kódují jako znaky plus (+). Tato funkce je vhodná při kódování řetězce, který se má použít jako query část URL jako příhodný způsob předání proměnných na další stránku:

Příklad 1. Příklad Urlencode()

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

Poznámka: pozor při předávání proměnných, které by mohly odpovídat HTML entitám. Věci jako &amp, &copy a &pound browser analyzuje a místo požadovaného jména proměnné použije odpovídající entitu. To je zřejmý problém, na který W3C upozorňuje už léta. Příručka je tady: http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2 PHP podporuje změnu oddělovače argumentů na středník doporučovaný W3C skrze .ini direktivu arg_separator. Bohužel, většina uživatelských programů neposílá data z formulářů v tomto formátu. Přenositelnější formou je použít jako oddělovač & místo &. Na to nemusíte měnit arg_separator. Nechte ho na &, ale kódujte URL pomocí **htmlentities()** (urlencode($data)).

Příklad 2. Příklad na urlencode/htmlentities()

```
echo '<A HREF="mycgi?foo=', htmlentities (urlencode ($userinput) ), '">';
```

Viz také **urldecode()**, **htmlentities()**, **rawurldecode()**, **rawurlencode()**.

LXXVIII. Variable Functions

doubleval (PHP 3, PHP 4)

Get double value of a variable

```
double doubleval (mixed var)
```

Returns the double (floating point) value of *var*.

Var may be any scalar type. You cannot use **doubleval()** on arrays or objects.

```
$var = '122.34343The';
$double_value_of_var = doubleval ($var);
print $double_value_of_var; // prints 122.34343
```

See also **intval()**, **strval()**, **settype()** and Type juggling.

empty (unknown)

Determine whether a variable is set

```
int empty (mixed var)
```

Returns false if *var* is set and has a non-empty or non-zero value; true otherwise.

```
$var = 0;

if (empty($var)) {  // evaluates true
    echo '$var is either 0 or not set at all';
}

if (!isset($var)) { // evaluates false
    echo '$var is not set at all';
}
```

Note that this is meaningless when used on anything which isn't a variable; i.e. **empty (addslashes ($name))** has no meaning since it would be checking whether something which isn't a variable is a variable with a false value.

See also **isset()** and **unset()**.

gettype (PHP 3, PHP 4)

Get the type of a variable

```
string gettype (mixed var)
```

Returns the type of the PHP variable *var*.

Possibles values for the returned string are:

- "boolean"
- "integer"
- "double"
- "string"

- "array"
- "object"
- "resource"
- "user function" (PHP 3 only, deprecated)
- "unknown type"

For PHP 4, you should use **function_exists()** and **method_exists()** to replace the prior usage of **gettype()** on a function.

See also **settype()**.

get_defined_vars (PHP 4 >= 4.0.4)

Returns an array of all defined variables

```
array get_defined_vars (void )
```

This function returns an multidimensional array containing a list of all defined variables, be them environment, server or user-defined variables.

```
$b = array(1,1,2,3,5,8);

$arr = get_defined_vars();

// print $b
print_r($arr["b"]);

// print path to the PHP interpreter (if used as a CGI)
// e.g. /usr/local/bin/php
echo $arr["_"];

// print the command-line paramaters if any
print_r($arr["argv"]);

// print all the server vars
print_r($arr["HTTP_SERVER_VARS"]);

// print all the available keys for the arrays of variables
print_r(array_keys(get_defined_vars()));
```

See also **get_defined_functions()**.

get_resource_type (PHP 4 >= 4.0.2)

Returns the resource type

```
string get_resource_type (resource $handle)
```

This function returns a string representing the type of the resource passed to it. If the paramater is not a valid resource, it generates an error.

```
$c = mysql_connect();
echo get_resource_type($c)."\n";
// prints: mysql link
```

```
$fp = fopen("foo","w");
echo get_resource_type($fp)."\n";
// prints: file

$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc)."\n";
// prints: domxml document
```

intval (PHP 3, PHP 4)

Get integer value of a variable

```
int intval (mixed var [, int base])
```

Returns the integer value of *var*, using the specified base for the conversion (the default is base 10).

Var may be any scalar type. You cannot use **intval()** on arrays or objects.

See also **doubleval()**, **strval()**, **settype()** and Type juggling.

is_array (PHP 3, PHP 4)

Finds whether a variable is an array

```
bool is_array (mixed var)
```

Returns true if *var* is an array, false otherwise.

See also **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, and **is_object()**.

is_bool (PHP 4 >= 4.0b4)

Finds out whether a variable is a boolean

```
bool is_bool  (mixed var)
```

Returns true if the *var* parameter is a boolean.

See also **is_array()**, **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, and **is_object()**.

is_double (PHP 3, PHP 4)

Finds whether a variable is a double

```
bool is_double (mixed var)
```

Returns true if *var* is a double, false otherwise.

See also **is_array()**, **is_bool()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, and **is_object()**.

is_float (PHP 3, PHP 4)

Finds whether a variable is a float

```
bool is_float (mixed var)
```

This function is an alias for **is_double()**.

See also **is_double()**, **is_bool()**, **is_real()**, **is_int()**, **is_integer()**, **is_string()**, **is_object()**, **is_array()**, and **is_long()**.

is_int (PHP 3, PHP 4)

Find whether a variable is an integer

```
bool is_int (mixed var)
```

This function is an alias for **is_long()**.

See also **is_bool()**, **is_double()**, **is_float()**, **is_integer()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, and **is_long()**.

is_integer (PHP 3, PHP 4)

Find whether a variable is an integer

```
bool is_integer (mixed var)
```

This function is an alias for **is_long()**.

See also **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, and **is_long()**.

is_long (PHP 3, PHP 4)

Finds whether a variable is an integer

```
bool is_long (mixed var)
```

Returns true if *var* is an integer (long), false otherwise.

See also **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_null (PHP 4 >= 4.0.4)

Finds whether a variable is null

```
bool is_null (mixed var)
```

Returns true if *var* is null, false otherwise.

See also **is_bool()**, **is_double()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_numeric (PHP 4 >= 4.0RC1)

Finds whether a variable is a number or a numeric string

```
bool is_numeric (mixed var)
```

Returns true if *var* is a number or a numeric string, false otherwise.

See also **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_object (PHP 3, PHP 4)

Finds whether a variable is an object

```
bool is_object (mixed var)
```

Returns true if *var* is an object, false otherwise.

See also **is_bool()**, **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_string()**, and **is_array()**.

is_real (PHP 3, PHP 4)

Finds whether a variable is a real

```
bool is_real (mixed var)
```

This function is an alias for **is_double()**.

See also **is_bool()**, **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_object()**, **is_string()**, and **is_array()**.

is_resource (PHP 4 >= 4.0b4)

Finds whether a variable is a resource

```
bool is_resource  (mixed var)
```

is_resource() returns true if the variable given by the *var* parameter is a resource, otherwise it returns false.

Resources are things like file or database result handles that are allocated and freed by internal PHP functions and that may need some cleanup when they are no longer in use but haven't been freed by user code.

is_scalar (PHP 4 CVS only)

Finds whether a variable is a scalar

```
bool is_scalar  (mixed var)
```

is_scalar() returns true if the variable given by the *var* parameter is a scalar, otherwise it returns false.

Scalar variables are those containing an integer, float, string or boolean. For example:

```
function show_var($var) {
    if (is_scalar($var))
```

```
        echo $var;
    else
        var_dump($var);
}

$pi = 3.1416;
$proteins = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

show_var($pi);
// prints: 3.1416

show_var($proteins)
// prints:
// array(3) {
//   [0]=>
//   string(10) "hemoglobin"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxin"
// }
```

Poznámka: This function was added to the CVS code after the release of PHP 4.0.4pl1

See also **is_bool()**, **is_double()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_string (PHP 3, PHP 4)

Finds whether a variable is a string

```
bool is_string (mixed var)
```

Returns true if *var* is a string, false otherwise.

See also **is_bool()**, **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_object()**, and **is_array()**.

isset (unknown)

Determine whether a variable is set

```
int isset (mixed var)
```

Returns true if *var* exists; false otherwise.

If a variable has been unset with **unset()**, it will no longer be **isset()**.

```
$a = "test";
echo isset ($a); // true
unset ($a);
echo isset ($a); // false
```

See also **empty()** and **unset()**.

print_r (PHP 4)

Prints human-readable information about a variable

```
void print_r (mixed expression)
```

This function displays information about the values of variables in a way that's readable by humans. If given a string, integer or double, the value itself will be printed. If given an array, values will be presented in a format that shows keys and elements. Similar notation is used for objects.

Compare **print_r()** to **var_dump()**.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
print_r ($a);
?>
```

> **Varování**
>
> This function will continue forever if given an array or object that contains a direct or indirect reference to itself or that contains an array or object on a deeper level that does so. This is especially true for `print_r($GLOBALS)`, as `$GLOBALS` is itself a global variable and contains a reference to itself as such.

serialize (PHP 3>= 3.0.5, PHP 4)

Generates a storable representation of a value

```
string serialize (mixed value)
```

Serialize() returns a string containing a byte-stream representation of *value* that can be stored anywhere.

This is useful for storing or passing PHP values around without losing their type and structure.

To make the serialized string into a PHP value again, use **unserialize()**. **Serialize()** handles the types integer, double, string, array (multidimensional) and object (object properties will be serialized, but methods are lost).

Příklad 1. Serialize() example

```
// $session_data contains a multi-dimensional array with session
// information for the current user.  We use serialize() to store
// it in a database at the end of the request.

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn,
      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
    /* Something went wrong.  Bitch, whine and moan. */
    }
}
```

settype (PHP 3, PHP 4)

Set the type of a variable

```
int settype (string var, string type)
```

Set the type of variable *var* to *type*.

Possibles values of *type* are:

- "integer"
- "double"
- "string"
- "array"
- "object"

Returns true if successful; otherwise returns false.

See also **gettype()**.

strval (PHP 3, PHP 4)

Get string value of a variable

```
string strval (mixed var)
```

Returns the string value of *var*.

var may be any scalar type. You cannot use **strval()** on arrays or objects.

See also **doubleval()**, **intval()**, **settype()** and Type juggling.

unserialize (PHP 3>= 3.0.5, PHP 4)

Creates a PHP value from a stored representation

```
mixed unserialize (string str)
```

unserialize() takes a single serialized variable (see **serialize()**) and converts it back into a PHP value. The converted value is returned, and can be an integer, double, string, array or object. If an object was serialized, its methods are not preserved in the returned value.

Příklad 1. Unserialize() example

```
// Here, we use unserialize() to load session data from a database
// into $session_data.    This example complements the one described
// with serialize().

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // if the execute or fetch fails, initialize to empty array
    $session_data = array();
} else {
```

```
    // we should now have the serialized data in $tmp[0].
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
    // something went wrong, initialize to empty array
    $session_data = array();
    }
}
```

unset (unknown)

Unset a given variable

```
void unset (mixed var [, mixed var [, ...]])
```

unset() destroys the specified variables. Note that in PHP 3, **unset()** will always return true (actually, the integer value 1). In PHP 4, however, **unset()** is no longer a true function: it is now a statement. As such no value is returned, and attempting to take the value of **unset()** results in a parse error.

Příklad 1. Unset() example

```
// destroy a single variable
unset ($foo);

// destroy a single element of an array
unset ($bar['quux']);

// destroy more than one variable
unset ($foo1, $foo2, $foo3);
```

The behavior of **unset()** inside of a function can vary depending on what type of variable you are attempting to destroy.

If a globalized variable is **unset()** inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before **unset()** was called.

```
function destroy_foo() {
    global $foo;
    unset($foo);
}

$foo = 'bar';
destroy_foo();
echo $foo;
```

The above example would output:

```
bar
```

If a variable that is PASSED BY REFERENCE is **unset()** inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before **unset()** was called.

```
function foo(&$bar) {
    unset($bar);
    $bar = "blah";
}
```

```
$bar = 'something';
echo "$bar\n";

foo($bar);
echo "$bar\n";
```

The above example would output:

```
something
something
```

If a static variable is **unset()** inside of a function, **unset()** unsets the reference to the static variable, rather than the static variable itself.

```
function foo() {
    static $a;
    $a++;
       echo "$a\n";

    unset($a);
}

foo();
foo();
foo();
```

The above example would output:

```
1
2
3
```

If you would like to **unset()** a global variable inside of a function, you can use the *$GLOBALS* array to do so:

```
function foo() {
    unset($GLOBALS['bar']);
}

$bar = "something";
foo();
```

Poznámka: Unset() is a language construct.

See also **isset()** and **empty()**.

var_dump (PHP 3>= 3.0.5, PHP 4)

Dumps information about a variable

```
void var_dump (mixed expression)
```

This function returns structured information about an expression that includes its type and value. Arrays are explored recursively with values indented to show structure.

Compare **var_dump()** to **print_r()**.

```
<pre>
<?php
    $a = array (1, 2, array ("a", "b", "c"));
    var_dump ($a);
?>
</pre>
```

LXXIX. WDDX funkce

Tyto funkce jsou určeny pro práci s WDDX (http://www.wddx.org/).

Pokud chcete používat WDDX, budete muset nainstalovat expat knihovnu (která je u Apache 1.3.7 a vyšších) a zkompilovat PHP s `-with-xml` a `-enable-wddx`.

Pozn.: všechny funkce které serializují proměnné používají první element pole k rozhodnutí jestli se toto pole serializuje do pole nebo struktury. Pokud má první element řetězec jako index, serializuje se do struktury, jinak do pole.

Příklad 1. Serializace jediné hodnoty

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

Tato ukázka vytvoří:

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Příklad 2. Použití inkrementálních paketů

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* Suppose $cities came from database */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Tato ukázka vytvoří:

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Serializovat jedinou hodnotu do WDDX paketu

```
string wddx_serialize_value (mixed var [, string comment])
```

wddx_serialize_value() se používá k vytvoření WDDX paketu z jediné dané hodnoty. Přijímá hodnotu obsaženou ve *var* a volitelný řetězec *comment*, který se použije v hlavičce paketu, a vrací WDDX paket.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Serializovat proměnné do WDDX paketu

```
string wddx_serialize_vars (mixed var_name [, mixed ...])
```

wddx_serialize_vars() se používá k vytvoření WDDX paketu se strukturou která obsahuje serializovanou reprezentaci předaných proměnných.

wddx_serialize_vars() přijímá proměnný počet argumentů, každý z nich může být buď řetězec obsahující název proměnné, nebo pole názvů proměnných, nebo jiné pole atd.

Příklad 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

Výše uvedená ukázka vytvoří:

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

wddx_packet_start (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Začít nový WDDX paket obsahující strukturu

```
int wddx_packet_start ([string comment])
```

wddx_packet_start() se používá k započetí nového WDDX paketu pro inkrementální přidávání proměnných. Přijímá volitelný řetězec *comment* a vrací ID packetu pro použití v dalších funkcích. Uvnitř paketu automaticky vytváří definici struktury která bude obsahovat přidané proměnné.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Ukončit WDDX paket se zadaným ID

```
string wddx_packet_end (int packet_id)
```

wddx_packet_end() ukončí WDDX paket určený argumentem *packet_id* a vrací řetězec obsahující tento paket.

wddx_add_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Přidat proměnné do WDDX paketu s určeným ID

```
wddx_add_vars (int packet_id, mixed name_var [, mixed ...])
```

wddx_add_vars() se používá k serializaci předaných proměnných a přidání výsledku do paketu specifikovaného *packet_id*. Proměnné určené k serializaci se udávají stejně jako u **wddx_serialize_vars()**.

wddx_deserialize (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Deserializovat WDDX paket

```
mixed wddx_deserialize (string packet)
```

wddx_deserialized() přijímá řetězec *packet* a deserializuje ho. Vrací výsledek, což může být řetězec, číslo, nebo pole. Pozn.: Struktury se deserializují do asociativních polí.

LXXX. XML parser functions

Introduction

About XML

XML (eXtensible Markup Language) is a data format for structured document interchange on the Web. It is a standard defined by The World Wide Web consortium (W3C). Information about XML and related technologies can be found at http://www.w3.org/XML/.

Installation

This extension uses expat, which can be found at http://www.jclark.com/xml/. The Makefile that comes with expat does not build a library by default, you can use this make rule for that:

```
libexpat.a: $(OBJS)
    ar -rc $@ $(OBJS)
    ranlib $@
```

A source RPM package of expat can be found at http://www.guardian.no/~ssb/phpxml.html.

Note that if you are using Apache-1.3.7 or later, you already have the required expat library. Simply configure PHP using `-with-xml` (without any additional path) and it will automatically use the expat library built into Apache.

On UNIX, run **configure** with the `-with-xml` option. The expat library should be installed somewhere your compiler can find it. If you compile PHP as a module for Apache 1.3.9 or later, PHP will automatically use the bundled expat library from Apache. You may need to set CPPFLAGS and LDFLAGS in your environment before running configure if you have installed expat somewhere exotic.

Build PHP. *Tada!* That should be it.

About This Extension

This PHP extension implements support for James Clark's expat in PHP. This toolkit lets you parse, but not validate, XML documents. It supports three source character encodings also provided by PHP: `US-ASCII`, `ISO-8859-1` and `UTF-8`. `UTF-16` is not supported.

This extension lets you create XML parsers and then define *handlers* for different XML events. Each XML parser also has a few parameters you can adjust.

The XML event handlers defined are:

Tabulka 1. Supported XML handlers

PHP function to set handler	Event description
xml_set_element_handler()	Element events are issued whenever the XML parser encounters start or end tags. There are separate handlers for start tags and end tags.
xml_set_character_data_handler()	Character data is roughly all the non-markup contents of XML documents, including whitespace between tags. Note that the XML parser does not add or remove any whitespace, it is up to the application (you) to decide whether whitespace is significant.

PHP function to set handler	Event description
xml_set_processing_instruction_handler()	PHP programmers should be familiar with processing instructions (PIs) already. <?php ?> is a processing instruction, where `php is called the "PI target". The handling of these are application-specific, except that all PI targets starting with "XML" are reserved.`
xml_set_default_handler()	What goes not to another handler goes to the default handler. You will get things like the XML and document type declarations in the default handler.
xml_set_unparsed_entity_decl_handler()	This handler will be called for declaration of an unparsed (NDATA) entity.
xml_set_notation_decl_handler()	This handler is called for declaration of a notation.
xml_set_external_entity_ref_handler()	This handler is called when the XML parser finds a reference to an external parsed general entity. This can be a reference to a file or URL, for example. See the external entity example for a demonstration.

Case Folding

The element handler functions may get their element names *case-folded*. Case-folding is defined by the XML standard as "a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents". In other words, when it comes to XML, case-folding simply means uppercasing.

By default, all the element names that are passed to the handler functions are case-folded. This behaviour can be queried and controlled per XML parser with the **xml_parser_get_option()** and **xml_parser_set_option()** functions, respectively.

Error Codes

The following constants are defined for XML error codes (as returned by **xml_parse()**):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING

XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Character Encoding

PHP's XML extension supports the Unicode (http://www.unicode.org/) character set through different *character encoding*s. There are two types of character encodings, *source encoding* and *target encoding*. PHP's internal representation of the document is always encoded with `UTF-8`.

Source encoding is done when an XML document is parsed. Upon creating an XML parser, a source encoding can be specified (this encoding can not be changed later in the XML parser's lifetime). The supported source encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`. The former two are single-byte encodings, which means that each character is represented by a single byte. `UTF-8` can encode characters composed by a variable number of bits (up to 21) in one to four bytes. The default source encoding used by PHP is `ISO-8859-1`.

Target encoding is done when PHP passes data to XML handler functions. When an XML parser is created, the target encoding is set to the same as the source encoding, but this may be changed at any point. The target encoding will affect character data as well as tag names and processing instruction targets.

If the XML parser encounters characters outside the range that its source encoding is capable of representing, it will return an error.

If PHP encounters characters in the parsed XML document that can not be represented in the chosen target encoding, the problem characters will be "demoted". Currently, this means that such characters are replaced by a question mark.

Some Examples

Here are some example PHP scripts parsing XML documents.

XML Element Structure Example

This first example displays the stucture of the start elements in a document with indentation.

Příklad 1. Show XML Element Structure

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]-;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
```

```
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML Tag Mapping Example

Příklad 2. Map XML to HTML

This example maps tags in an XML document directly to HTML tags. Elements not found in the "map array" are ignored. Of course, this example will only work with a specific XML document type.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML External Entity Example

This example highlights XML code. It illustrates how to use an external entity reference handler to include and parse other documents, as well as how PIs can be processed, and a way of determining "trust" for PIs containing code.

XML documents that can be used for this example are found below the example (`xmltest.xml` and `xmltest2.xml`.)

Příklad 3. External Entity Example

```
$file = "xmltest.xml";

function trustedFile($file) {
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
```

```
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}


function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);
```

```
?>
```

Příklad 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!- this is a comment ->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

This file is included from `xmltest.xml`:

Příklad 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

xml_parser_create (PHP 3>= 3.0.6, PHP 4)

create an XML parser

```
int xml_parser_create ([string encoding])
```

encoding (optional)

Which character encoding the parser should use. The following character encodings are supported:

`ISO-8859-1 (default)`

`US-ASCII`

`UTF-8`

This function creates an XML parser and returns a handle for use by other XML functions. Returns `false` on failure.

xml_set_object (PHP 4 >= 4.0b4)

Use XML Parser within an object

```
void xml_set_object (int parser, object &object)
```

This function allows to use *parser* inside *object*. All callback functions could be set with **xml_set_element_handler()** etc and assumed to be methods of *object*.

```
<?php
class xml  {
var $parser;

function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}

function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}

} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("<A ID=\"hallo\">PHP</A>");
?>
```

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4)

set up start and end element handlers

```
int xml_set_element_handler  (int parser, string startElementHandler, string
endElementHandler)
```

Sets the element handler functions for the XML parser *parser*. *startElementHandler* and *endElementHandler* are strings containing the names of functions that must exist when **xml_parse()** is called for *parser*.

The function named by *startElementHandler* must accept three parameters:

```
startElementHandler (int parser, string name, array attribs)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

attribs

The third parameter, *attribs*, contains an associative array with the element's attributes (if any). The keys of this array are the attribute names, the values are the attribute values. Attribute names are case-folded on the same criteria as element names. Attribute values are *not* case-folded.

The original order of the attributes can be retrieved by walking through *attribs* the normal way, using **each()**. The first key in the array was the first attribute, and so on.

The function named by *endElementHandler* must accept two parameters:

```
endElementHandler (int parser, string name)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handlers are set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4)

set up character data handler

```
int xml_set_character_data_handler  (int parser, string handler)
```

Sets the character data handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters:

```
handler (int parser, string data)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data as a string.

If a handler function is set to an empty string, or false, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4)

Set up processing instruction (PI) handler

```
int xml_set_processing_instruction_handler  (int parser, string handler)
```

Sets the processing instruction (PI) handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A processing instruction has the following format:

```
<?
       target
       data?>
```

You can put PHP code into such a tag, but be aware of one limitation: in an XML PI, the PI end tag (?>) can not be quoted, so this character sequence should not appear in the PHP code you embed with PIs in XML documents. If it does, the rest of the PHP code, as well as the "real" PI end tag, will be treated as character data.

The function named by *handler* must accept three parameters:

```
handler (int parser, string target, string data)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

target

The second parameter, *target*, contains the PI target.

data

The third parameter, *data*, contains the PI data.

If a handler function is set to an empty string, or false, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4)

set up default handler

```
int xml_set_default_handler  (int parser, string handler)
```

Sets the default handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters:

```
handler (int parser, string data)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data. This may be the XML declaration, document type declaration, entities or other data for which no other handler exists.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4)

Set up unparsed entity declaration handler

```
int xml_set_unparsed_entity_decl_handler  (int parser, string handler)
```

Sets the unparsed entity declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

This handler will be called if the XML parser encounters an external entity declaration with an NDATA declaration, like the following:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

See section 4.2.2 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) for the definition of notation declared external entities.

The function named by *handler* must accept six parameters:

```
handler (int parser, string entityName, string base, string systemId, string publicId,
string notationName)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

entityName

The name of the entity that is about to be defined.

base

This is the base for resolving the system identifier (*systemId*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

System identifier for the external entity.

publicId

Public identifier for the external entity.

notationName

Name of the notation of this entity (see **xml_set_notation_decl_handler()**).

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4)

set up notation declaration handler

```
int xml_set_notation_decl_handler  (int parser, string handler)
```

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A notation declaration is part of the document's DTD and has the following format:

```
<!NOTATION
     name {systemId |
     publicId}>
```

See section 4.7 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#Notations) for the definition of notation declarations.

The function named by *handler* must accept five parameters:

```
handler (int parser, string notationName, string base, string systemId, string
publicId)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

notationName

This is the notation's *name*, as per the notation format described above.

base

This is the base for resolving the system identifier (*systemId*) of the notation declaration. Currently this parameter will always be set to an empty string.

systemId

System identifier of the external notation declaration.

publicId

Public identifier of the external notation declaration.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4)

set up external entity reference handler

```
int xml_set_external_entity_ref_handler  (int parser, string handler)
```

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept five parameters, and should return an integer value. If the value returned from the handler is false (which it will be if no value is returned), the XML parser will stop parsing and **xml_get_error_code()** will return XML_ERROR_EXTERNAL_ENTITY_HANDLING.

```
int handler (int parser, string openEntityNames, string base, string systemId, string
publicId)
```

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

openEntityNames

The second parameter, *openEntityNames*, is a space-separated list of the names of the entities that are open for the parse of this entity (including the name of the referenced entity).

base

This is the base for resolving the system identifier (*systemid*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

The fourth parameter, *systemId*, is the system identifier as specified in the entity declaration.

publicId

The fifth parameter, *publicId*, is the public identifier as specified in the entity declaration, or an empty string if none was specified; the whitespace in the public identifier will have been normalized as required by the XML spec.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers. See **xml_set_object()** for using the XML parser within an object.

xml_parse (PHP 3>= 3.0.6, PHP 4)

start parsing an XML document

```
int xml_parse (int parser, string data [, int isFinal])
```

parser

A reference to the XML parser to use.

data

Chunk of data to parse. A document may be parsed piece-wise by calling **xml_parse()** several times with new data, as long as the *isFinal* parameter is set and true when the last data is parsed.

isFinal (optional)

If set and true, *data* is the last piece of data sent in this parse.

When the XML document is parsed, the handlers for the configured events are called as many times as necessary, after which this function returns true or false.

True is returned if the parse was successful, false if it was not successful, or if *parser* does not refer to a valid parser. For unsuccessful parses, error information can be retrieved with **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** and **xml_get_current_byte_index()**.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4)

get XML parser error code

```
int xml_get_error_code (int parser)
```

parser

A reference to the XML parser to get error code from.

This function returns false if *parser* does not refer to a valid parser, or else it returns one of the error codes listed in the error codes section.

xml_error_string (PHP 3>= 3.0.6, PHP 4)

get XML parser error string

```
string xml_error_string (int code)
```

code

An error code from **xml_get_error_code()**.

Returns a string with a textual description of the error code *code*, or false if no description was found.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4)

get current line number for an XML parser

```
int xml_get_current_line_number  (int parser)
```

parser

A reference to the XML parser to get line number from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which line the parser is currently at in its data buffer.

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4)

Get current column number for an XML parser

```
int xml_get_current_column_number  (int parser)
```

parser

A reference to the XML parser to get column number from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which column on the current line (as given by **xml_get_current_line_number()**) the parser is currently at.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4)

get current byte index for an XML parser

```
int xml_get_current_byte_index  (int parser)
```

parser

A reference to the XML parser to get byte index from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which byte index the parser is currently at in its data buffer (starting at 0).

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4)

Parse XML data into an array structure

```
int xml_parse_into_struct (int parser, string data, array &values, array &index)
```

This function parses an XML file into 2 parallel array structures, one (*index*) containing pointers to the location of the appropriate values in the *values* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple `note` tag embeded inside a `para` tag, and then we parse this an print out the structures generated:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed

in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let’s consider the following XML file representing a small database of aminoacids information:

Příklad 1. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Příklad 2. parsemoldb.php - parses moldb.xml into and array of molecular objects

```
<?php

class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);

    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
```

```
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable `$db` contains an array of `AminoAcid` objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )

    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
            [type] => charged
        )

)
```

xml_parser_free (PHP 3>= 3.0.6, PHP 4)

Free an XML parser

```
string xml_parser_free (int parser)
```

parser

A reference to the XML parser to free.

This function returns false if *parser* does not refer to a valid parser, or else it frees the parser and returns true.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4)

set options in an XML parser

```
int xml_parser_set_option (int parser, int option, mixed value)
```

parser

A reference to the XML parser to set an option in.

option

Which option to set. See below.

value

The option's new value.

This function returns false if *parser* does not refer to a valid parser, or if the option could not be set. Else the option is set and true is returned.

The following options are available:

Tabulka 1. XML parser options

Option constant	Data type	Description
XML_OPTION_CASE_FOLDING	integer	Controls whether case-folding is enabled for this XML parser. Enabled by default.
XML_OPTION_TARGET_ENCODING	string	Sets which target encoding to use in this XML parser. By default, it is set to the same as the source encoding used by **xml_parser_create(). Supported target encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`.**

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4)

get options from an XML parser

```
mixed xml_parser_get_option  (int parser, int option)
```

parser

A reference to the XML parser to get an option from.

option

Which option to fetch. See **xml_parser_set_option()** for a list of options.

This function returns false if *parser* does not refer to a valid parser, or if the option could not be set. Else the option's value is returned.

See **xml_parser_set_option()** for the list of options.

utf8_decode (PHP 3>= 3.0.6, PHP 4)

Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.

```
string utf8_decode (string data)
```

This function decodes *data*, assumed to be UTF-8 encoded, to ISO-8859-1.

See **utf8_encode()** for an explaination of UTF-8 encoding.

utf8_encode (PHP 3>= 3.0.6, PHP 4)

encodes an ISO-8859-1 string to UTF-8

```
string utf8_encode (string data)
```

This function encodes the string *data* to UTF-8, and returns the encoded version. UTF-8 is a standard mechanism used by Unicodefor encoding *wide character* values into a byte stream. UTF-8 is transparent to plain ASCII characters, is self-synchronized (meaning it is possible for a program to figure out where in the bytestream characters start) and can be used with normal string comparison functions for sorting and such. PHP encodes UTF-8 characters in up to four bytes, like this:

Tabulka 1. UTF-8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Each *b* represents a bit that can be used to store character data.

LXXXI. XSLT funkce

Úvod

O XSLT and Sablotronu

XSLT (Extensible Stylesheet Language (XSL) Transformations) je jazyk pro transformaci XML dokumentů do jiných XML dokumentů. Je to standard definovaný The World Wide Web konsorciem (W3C). Imformace o XSLT a souvisejících technologiích jsou dostupné na http://www.w3.org/TR/xslt.

Instalace

Tato extenze využívá Sabloton a expat, které jsou dostupné na http://www.gingerall.com/, a to jak binární soubory tak zdrojové kódy.

Na UNIXu spusťte **configure** s `-with-sablot`. Sablotron knihovna by měla být nainstalována na nějakém místě, kde ji váš kompilátor může najít.

O této extenzi

Tato PHP extenze implementuje podporu Sablotron od Ginger Alliance v PHP. Tato nástrojů vám umožňuje transformovat XML dokumenty na jiné dokumenty, včetně nových XML dokumentů, ale také do XML a jiných cílových formátů. V podstatě poskytuje standardizovaný a přenosný systém šablon oddělující obsah a design websajty.

xslt_closelog (PHP 4 >= 4.0.3)

Smazat log dané instance Sablotronu

```
bool xslt_closelog (resource xh)
```

xh

Reference na XSLT parser.

Pokud *parser* neodkazuje na platný parser, nebo pokud zavření logu selže, vrací `false`, jinak vrací `true`

xslt_create (PHP 4 >= 4.0.3)

Vytvořit nový XSL procesor

```
resource xslt_create(void);
```

Tato funkce vrací handle nového XSL procesoru. Tento handle je potřeba ve všech následných voláních XSL funkcí.

xslt_errno (PHP 4 >= 4.0.3)

Vrátit číslo současné chyby

```
int xslt_errno ([int xh])
```

Vrací číslo současné chyby daného XSL procesoru. Pokud nedostane handle, vrací číslo poslední chyby bez ohledu na její výskyt.

xslt_error (PHP 4 >= 4.0.3)

Vrátit text poslední chyby

```
mixed xslt_error ([int xh])
```

Vrací text současné chyby daného XSL procesoru. Pokud nedostane handle, vrací text poslední chyby bez ohledu na její výskyt.

xslt_fetch_result (PHP 4 >= 4.0.3)

Získat (pojmenovaný) výstupní buffer

```
string xslt_fetch_result ([int xh string result_name])
```

Vrací výstupní buffer XSLT procesoru identifikovaného předaným handle. Pokud nedostane jméno výstupního bufferu, vrací buffer pojmenovaný "/_result".

xslt_free (PHP 4 >= 4.0.3)

Uvolnit XSLT procesor

```
void xslt_free (resource xh)
```

Uvolní XSLT procesor identifikovaný předaným handle.

xslt_openlog (PHP 4 >= 4.0.3)

Určit log pro zprávy XSLT procesoru

```
bool xslt_openlog ([resource xh string logfile int loglevel])
```

Určí log soubor, do kterého má XSLT procesor umisťovat všechny chybové zprávy.

xslt_output_begintransform (PHP 4 >= 4.0.3)

Začít XSLT transformaci výstupu

```
void xslt_output_begintransform  (string xslt_filename)
```

Tato funkce začne výstupní transformaci vašich dat. Od okamžiku, kdy zavoláte **xslt_output_begintransform()** až do chvíle kdy zavoláte **xslt_output_endtransform()** bude všechen výstup transformován XSLT stylesheetem udaným v prvním argumentu.

Příklad 1. Transformace výstupu XSLT stylesheetem pomocí DOM-XML funkcí pro generování XML

```
<?php

$xsl_file = "article.xsl";
xslt_output_begintransform($xsl_file);

$doc = new_xmldoc('1.0');
$article = $doc->new_root('article');

$article->new_child('title', 'The History of South Tyrol');
$article->new_child('author', 'Sterling Hughes');
$article->new_child('body', 'Back after WWI, Italy gained South Tyrol from
                             Austria.  Since that point nothing interesting has
                             happened');

echo $doc->dumpmem();

xslt_output_endtransform();
```

xslt_output_endtransform (PHP 4 >= 4.0.3)

Ukončit výstupní transformaci začatou pomocí xslt_output_begintransform

```
void xslt_output_endtransform (void);
```

xslt_output_endtransform() ukončí výstupní transformaci začatou funkcí **xslt_output_begintransform()**. Pokud chcete vidět výsledky výstupní transformace, musíte tuto funkci zavolat.

xslt_process (PHP 4 >= 4.0.3)

Transformovat XML data řetězcem obsahujícím XSL data

```
bool xslt_process  (string xsl_data string xml_data string result)
```

xslt_process() přijímá jako první argument řetězec obsahující XSLT stylesheet, jako druhý argument řetězec obsahující XML data, která chcete transformovat, a jako třetí argument řetězec obsahující výsledky transformace. **xslt_process()** vrací `true` při úspěchu a `false` při selhání. Číslo a text chyby případně vzniklé při transformaci můžete získat pomocí **xslt_errno()** a **xslt_error()** funkcí.

Příklad 1. Použití xslt_process() k transformaci tří řetězců

```
<?php

$xslData = ’
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="article">
    <table border="1" cellpadding="2" cellspacing="1">
        <tr>
            <td width="20%">

            </title>
            <td width="80%">
                <h2><xsl:value-of select="title"></h2>
                <h3><xsl:value-of select="author"></h3>
                <br>

                <xsl:value-of select="body">
            </td>
        </tr>
    </table>
</xsl:template>

</xsl:stylesheet>’;

$xmlData = ’
<?xml version="1.0"?>
<article>
    <title>Learning German</title>
    <author>Sterling Hughes</author>
    <body>
      Essential phrases:
      <br>
      <br>
      Komme sie mir sagen, woe die toilette es?<br>
      Eine grande beer bitte!<br>
      Noch einem bitte.<br>
    </body>
</article>’;

if (xslt_process($xslData, $xmlData, $result))
{
    echo "Here is the brilliant in-depth article on learning";
    echo " German: ";
    echo "<br>\n<br>";
```

```
    echo $result;
}
else
{
    echo "There was an error that occurred in the XSL transformace...\n";
    echo "\tError number: " . xslt_errno() . "\n";
    echo "\tError string: " . xslt_error() . "\n";
    exit;
}
?>
```

xslt_run (PHP 4 >= 4.0.3)

Aplikovat na soubor XSLT stylesheet

`bool` **`xslt_run`** `([`*`resource xh string xslt_file string xml_data_file string result array xslt_params array xslt_args`*`]])`

Zpracuje Zpracovat *`string xml_data_file`* aplikací *`string xslt_file`* stylesheetu. Tento stylesheet má přístup ke *`xslt_params`* a nastartuje se procesor s *`xslt_args`*. Výsledek XSLT transformace se umístí do pojmenovaného bufferu (defaultně "/_result").

xslt_set_sax_handler (PHP 4 >= 4.0.3)

Určit SAX handlery XSLT procesoru

`bool` **`xslt_set_sax_handler`** `(`*`resource xh array handlers`*`)`

Určit SAX handlery pro handle určený v *`xh`*.

xslt_transform (PHP 4 >= 4.0.3)

Provést XSLT transformaci

`bool` **`xslt_transform`** `(`*`string xsl string xml string result string params string args string resultBuffer`*`)`

xslt_transform() poskytuje interface k pokročilejším vlastnostem Sablotronu bez nutnosti použít resource API.

LXXXII. YAZ functions

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 protocol for information retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

YAZ is available at http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at http://www.indexdata.dk/phpyaz/.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and initialize phase steps in most cases.

Installation

Compile YAZ and install it. Build PHP with your favourite modules and add option –with-yaz. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure -prefix=/usr
make
make install
cd ../php-4.0.X
./configure -with-yaz=/usr/bin
make
make install
```

Example

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

Příklad 1. Parallel searching using YAZ()

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in in array host.

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
```

```
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

yaz_addinfo (PHP 4 >= 4.0.1)

Returns additional error information

```
int yaz_addinfo (int id)
```

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_close (PHP 4 >= 4.0.1)

Closes a YAZ connection

```
int yaz_close (int id)
```

Closes the Z-association given by *id*. The *id* is a target ID as returned by a previous **yaz_connect()** command.

yaz_connect (PHP 4 >= 4.0.1)

Prepares for a connection and Z-association to a Z39.50 target.

```
int yaz_connect (string zurl [, string authentication])
```

This function returns a positive ID on success; zero on failure.

Yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when **yaz_wait()** is called.

yaz_errno (PHP 4 >= 4.0.1)

Returns error number

```
int yaz_errno (int id)
```

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors (such as when the target closed connection, etc).

yaz_errno() should be called after network activity for each target - (after **yaz_wait()** returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4 >= 4.0.1)

Returns error description

```
string yaz_error (int id)
```

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns an english text message corresponding to the last error number as returned by **yaz_errno()**.

yaz_hits (PHP 4 >= 4.0.1)

Returns number of hits for last search

```
int yaz_hits (int id)
```

Yaz_hits() returns number of hits for last search.

yaz_element (PHP 4 >= 4.0.1)

Specifies Element-Set Name for retrieval

```
int yaz_element (int id, string elementset)
```

This function is used in conjunction with **yaz_search()** and **yaz_present()** to specify the element set name for records to be retrieved. Most servers support F (full) and B (brief).

Returns true on success; false on error.

yaz_database (PHP 4 CVS only)

Specifies the databases within a session

```
int yaz_database (int id, string databases)
```

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to **yaz_connect()**. Multiple databases are separated by a plus sign +.

This function allows you to use different sets of databases within a session.

Returns true on success; false on error.

yaz_range (PHP 4 >= 4.0.1)

Specifies the maximum number of records to retrieve

```
int yaz_range (int id, int start, int number)
```

This function is used in conjunction with **yaz_search()** to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only **yaz_search()**) start is set to 1 and number is set to 10.

Returns true on success; false on error.

yaz_record (PHP 4 >= 4.0.1)

Returns a record

```
int yaz_record (int id, int pos, string type)
```

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_search (PHP 4 >= 4.0.1)

Prepares for a search

```
int yaz_search (int id, string type, string query)
```

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument specifies a Type-1 query (RPN). Like **yaz_connect()** this function is non-blocking and only prepares for a search to be executed later when **yaz_wait()** is called.

The RPN query is a textual represenation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operater precedes the operands. The query string is a sequence of tokens where white space is ignored is ignored unless surrounded by double quotes. Tokens beginning with an at-character (@) are considered operators, otherwise they are treated as search terms.

Tabulka 1. RPN Operators

Syntax	Description
`@and query1 query2`	intersection of query1 and query2
`@or query1 query2`	union of query1 and query2
`@not query1 query2`	query1 and not query2
`@set name`	result set reference
`@attrset set query`	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
`@attr set type=value query`	applies attribute to query. The type and value are integers specifying the attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

The following illustrates valid query constructions:

```
computer
```

Matches documents where "computer" occur. No attributes are specified.

```
"donald knuth"
```

Matches documents where "donald knuth" occur.

```
@attr 1=4 art
```

Attribute type is 1 (Bib-1 use), attribute value is 4 Title), so this should match documents where "art" occur in the title.

```
@attrset gils @and @attr 1=4 art @attr 1=1003 "donald knuth"
```

The query as a whole uses the GILS attributeset. The query matches documents where "art" occur in the title and in which "donald knuth" occur in the author.

yaz_present (PHP 4 CVS only)

Prepares for retrieval (Z39.50 present).

```
int yaz_present(void);
```

This function prepares for retrieval of records after a successful search. The **yaz_range()** should be called prior to this function to specify the range of records to be retrieved.

yaz_syntax (PHP 4 >= 4.0.1)

Specifies the preferred record syntax for retrieval.

```
int yaz_syntax (int id, string syntax)
```

The syntax is specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.5.10`) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.). This function is used in conjunction with **yaz_search()** and **yaz_present()** to specify the preferred record syntax for retrieval.

yaz_scan (PHP 4 CVS only)

Prepares for a scan

```
int yaz_scan (int id, string type, string startterm [, array flags])
```

This function prepares for a Z39.50 Scan Request. Argument *`id`* specifies target ID. Starting term point for the scan is given by *`startterm`*. The form in which is the starting term is specified is given by *`type`*. Currently type `rpn` is supported. The optional *`flags`* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: `number` (number of terms requested), `position` (preferred position of term) and `stepSize` (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, **yaz_wait()** must be called. Upon completion of **yaz_wait()** call **yaz_error()** and **yaz_scan_result()** to handle the response.

The syntax of *`startterm`* is similar to the RPN query as described in **yaz_search()**. The startterm consists of zero or more `@attr`-operator specifications, then followed by exactly one token.

Příklad 1. PHP function that scans titles

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo 'Scan ok; ';
    $ar = yaz_scan_result($id, &$options);
    while(list($key,$val)=each($options)) {
       echo "$key = $val  ";
    }
    echo '<br><table><tr><td>';
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
      echo $tcount;
      echo "</td></tr>";
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br>";
  }
}
```

yaz_scan_result (PHP 4 CVS only)

Returns Scan Response result

```
array yaz_scan_result (int id [, array & result])
```

Given a target ID this function returns and array with terms as received from the target in the last Scan Response. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *result* is given it will be modified to hold additional information taken from the Scan Response: `number` (number of entries returned), `stepsize` (Step-size), `position` (position of term), `status` (Scan Status).

yaz_ccl_conf (PHP 4 CVS only)

Configure CCL parser

```
int yaz_ccl_conf (int id, array config)
```

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the **yaz_ccl_parse()** function. Each index of the array *config* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space.

Příklad 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is a target ID.

```
$field["ti"] = "1=4";
$field["au"] = "1=1";
$field["isbn"] = "1=7";
yaz_ccl_conf($id,$field);
```

yaz_ccl_parse (PHP 4 CVS only)

Invoke CCL Parser

```
int yaz_ccl_parse (int id, string query, array &result)
```

This function invokes the CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the **yaz_search()** function to perform a search. To define a set of valid CCL fields call **yaz_ccl_conf()** prior to this function. If the supplied *query* was successfully converted to RPN, this function returns true, and the index `rpn` of the supplied array *result* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns false and three indexes are set in the resulting array to indicate the cause of failure: `errorcode`CCL error code (integer), `errorstring`CCL error string, and `errorpos`approximate position in query of failure (integer is character position).

yaz_itemorder (PHP 4 CVS only)

Prepares for Z39.50 Item Order with an ILL-Request package

```
int yaz_itemorder (array args)
```

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) and the specification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED

requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_wait (PHP 4 >= 4.0.1)

Wait for Z39.50 requests to complete

```
int yaz_wait(void);
```

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions **yaz_connect()**, **yaz_search()**, **yaz_present()**, **yaz_scan()** and **yaz_itemorder()**. **yaz_wait()** returns when all targets have either completed all requests or aborted (in case of errors).

LXXXIII. Funkce pro práci s YP/NIS

NIS (dříve Yellow Pages) umožňuje síťovou správu důležitých administrativních souborů (např. soubory s hesly). Více informací viz NIS man stránka a Introduction to YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html). Existuje také kniha Managing NFS and NIS (http://www.oreilly.com/catalog/nfs/noframes.html) od Hala Sterna.

Pokud chcete tyto funkce zprovoznit, musíte PHP zkonfigurovat s `-with-yp`(PHP 3) nebo `-enable-yp`(PHP 4).

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4)

Zjistit defaultní NIS doménu stroje

```
int yp_get_default_domain (void )
```

yp_get_default_domain() vrací defaultní doménu uzlu nebo `false`. Dá se používat jako argument domény v následných voláních NIS.

NIS doména se dá popsat jako skupina map NIS. Každý server, který potřebuje vyhledat informace se připojí k určité doméně. Detailnější informace viz dokumentace zmíněná v úvodu.

Příklad 1. Ukázka defaultní domény

```
<?php
$domain = yp_get_default_domain();
echo "Defaultní NIS doména je: " . $domain;
?>
```

yp_order (PHP 3>= 3.0.7, PHP 4)

Returns the order number for a map

```
int yp_order (string domain, string map)
```

Yp_order() vrací pořadové číslo mapy nebo `false`.

Příklad 1. Ukázka NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Pořadové číslo této mapy je: " . $order;
?>
```

Viz také: **yp-get-default-domain()**.

yp_master (PHP 3>= 3.0.7, PHP 4)

Zjistit název master NIS serveru mapy

```
string yp_master (string domain, string map)
```

yp_master() vrací název master NIS serveru určité mapy.

Příklad 1. Ukázka NIS masteru

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master této mapy je: " . $master;
?>
```

Viz také **yp-get-default-domain()**.

yp_match (PHP 3>= 3.0.7, PHP 4)

Vrátit odpovídající záznam

```
string yp_match (string domain, string map, string key)
```

yp_match() vrací hodnotu asociovanou v dané mapě s předaným klíčem nebo `false`. Klíč musí být dán přesně.

Příklad 1. Ukázka NIS match

```
<?php
$entry = yp_match ($domain, "passwd.byname", "joe");
echo "Odpovídající záznam je: " . $entry;
?>
```

V tomto případě by to mohlo být: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

Viz také **yp-get-default-domain()**

yp_first (PHP 3>= 3.0.7, PHP 4)

Vrátit první klíč/hodnota pár mapy

```
array yp_first (string domain, string map)
```

Yp_first() vrací první klíč/hodnota pár dané mapy v dané doméně, nebo `false`.

Příklad 1. Ukázka NIS first

```
<?php
    $entry = yp_first($domain, "passwd.byname");
    $key = key($entry);
    echo "První záznam v této mapě má klíč " . $key
          . " a hodnotu " . $entry[$key];
?>
```

Viz také **yp-get-default-domain()**

yp_next (PHP 3>= 3.0.7, PHP 4)

Vrátit další klíč/hodnota pár mapy

```
array yp_next (string domain, string map, string key)
```

yp_next() vrací další klíč/hodnota pár v mapě po daném klíči, nebo `false`.

Příklad 1. Ukázka NIS next

```
<?php
$entry = yp_next ($domain, "passwd.byname", "joe");

if (!$entry) {
    echo yp_errno() . ": " . yp_err_string();
}

$key = key ($entry);

echo "Položka následující po joe má klíč " . $key
      . " a hodnotu " . $entry[$key];
?>
```

Viz také **yp-get-default-domain()**.

LXXXIV. Zlib Compression Functions

This module uses the functions of zlib (http://www.info-zip.org/pub/infozip/zlib/) by Jean-loup Gailly and Mark Adler to transparently read and write gzip (.gz) compressed files. You have to use a zlib version >= 1.0.9 with this module.

This module contains versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

> **Poznámka:** The current CVS version 4.0.4-dev introduces a fopen-wrapper for .gz-files, so that you can use a special 'zlib:' URL to access compressed files transparently using the normal f*() file access functions if you prepend the filename or path with a 'zlib:' prefix when calling **fopen()**.
>
> This feature requires a C runtime library that provides the `fopencookie()` function. To my current knowledge the GNU libc is the only library that provides this feature.

Small code example

Opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Příklad 1. Small Zlib Example

```
<?php

$filename = tempnam ('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
print gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);

print "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
        echo "Error with zlib functions!";
}
unlink($filename);
print "</pre>\n</h1></body>\n</html>\n";

?>
```

gzclose (PHP 3, PHP 4)

Close an open gz-file pointer

```
int gzclose (int zp)
```

The gz-file pointed to by zp is closed.

Returns true on success and false on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzeof (PHP 3, PHP 4)

Test for end-of-file on a gz-file pointer

```
int gzeof (int zp)
```

Returns true if the gz-file pointer is at EOF or an error occurs; otherwise returns false.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzfile (PHP 3, PHP 4)

Read entire gz-file into an array

```
array gzfile (string filename [, int use_include_path])
```

Identical to **readgzfile()**, except that **gzfile()** returns the file in an array.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also **readgzfile()**, and **gzopen()**.

gzgetc (PHP 3, PHP 4)

Get character from gz-file pointer

```
string gzgetc (int zp)
```

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns FALSE on EOF (as does **gzeof()**).

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, and **gzgets()**.

gzgets (PHP 3, PHP 4)

Get line from file pointer

```
string gzgets (int zp, int length)
```

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, **gzgetc()**, and **fgets()**.

gzgetss (PHP 3, PHP 4)

Get line from gz-file pointer and strip HTML tags

```
string gzgetss (int zp, int length [, string allowable_tags])
```

Identical to **gzgets()**, except that **gzgetss()** attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Poznámka: `Allowable_tags` was added in PHP 3.0.13, PHP4B3.

See also **gzgets()**, **gzopen()**, and **strip_tags()**.

gzopen (PHP 3, PHP 4)

Open gz-file

```
int gzopen (string filename, string mode [, int use_include_path])
```

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in **fopen()** ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

Gzopen() can be used to read a file which is not in gzip format; in this case **gzread()** will directly read from the file without decompression.

Gzopen() returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns false.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Příklad 1. Gzopen() Example

```
$fp = gzopen ("/tmp/file.gz", "r");
```

See also **gzclose()**.

gzpassthru (PHP 3, PHP 4)

Output all remaining data on a gz-file pointer

```
int gzpassthru (int zp)
```

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs (PHP 3, PHP 4)

Write to a gz-file pointer

```
int gzputs (int zp, string str [, int length])
```

Gzputs() is an alias to **gzwrite()**, and is identical in every way.

gzread (PHP 3, PHP 4)

Binary-safe gz-file read

```
string gzread (int zp, int length)
```

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen ($filename, "r");
$contents = gzread ($zd, 10000);
gzclose ($zd);
```

See also **gzwrite()**, **gzopen()**, **gzgets()**, **gzgetss()**, **gzfile()**, and **gzpassthru()**.

gzrewind (PHP 3, PHP 4)

Rewind the position of a gz-file pointer

```
int gzrewind (int zp)
```

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzseek()** and **gztell()**.

gzseek (PHP 3, PHP 4)

Seek on a gz-file pointer

```
int gzseek (int zp, int offset)
```

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also **gztell()** and **gzrewind()**.

gztell (PHP 3, PHP 4)

Tell gz-file pointer read/write position

```
int gztell (int zp)
```

Returns the position of the file pointer referenced by *zp*; i.e., its offset into the file stream.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, **gzseek()** and **gzrewind()**.

gzwrite (PHP 3, PHP 4)

Binary-safe gz-file write

```
int gzwrite (int zp, string string [, int length])
```

Gzwrite() writes the contents of *string* to the gz-file stream pointed to by *zp*. If the *length* argument is given, writing will stop after *length* (uncompressed) bytes have been written or the end of *string* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also **gzread()**, **gzopen()**, and **gzputs()**.

readgzfile (PHP 3, PHP 4)

Output a gz-file

```
int readgzfile (string filename [, int use_include_path])
```

Reads a file, decompresses it and writes it to standard output.

Readgzfile() can be used to read a file which is not in gzip format; in this case **readgzfile()** will directly read from the file without decompression.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, false is returned and unless the function was called as `@readgzfile`, an error message is printed.

The file *filename* will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also **gzpassthru()**, **gzfile()**, and **gzopen()**.

gzcompress (PHP 4 >= 4.0.1)

Compress a string

```
string gzcompress (string data [, int level])
```

This function returns a compressed version of the input *data* using the ZLIB data format, or false if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1950.txt)" (RFC 1950).

> **Poznámka:** This is *not* the same as gzip compression, which includes some header data. See **gzencode()** for gzip compression.

See also **gzdeflate()**, **gzinflate()**, **gzuncompress()**, **gzencode()**.

gzuncompress (PHP 4 >= 4.0.1)

Uncompress a deflated string

```
string gzuncompress (string data [, int length])
```

This function takes *data* compressed by **gzcompress()** and returns the original uncompressed data or false on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also **gzdeflate()**, **gzinflate()**, **gzcompress()**, **gzencode()**.

gzdeflate (PHP 4 >= 4.0.4)

Deflate a string

```
string gzdeflate (string data [, int level])
```

This function returns a compressed version of the input *data* using the DEFLATE data format, or false if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1951.txt)" (RFC 1951).

See also **gzinflate()**, **gzcompress()**, **gzuncompress()**, **gzencode()**.

gzinflate (PHP 4 >= 4.0.4)

Inflate a deflated string

```
string gzinflate (string data [, int length])
```

This function takes *data* compressed by **gzdeflate()** and returns the original uncompressed data or false on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also **gzcompress()**. **gzuncompress()**, **gzdeflate()**, **gzencode()**.

gzencode (PHP 4 >= 4.0.4)

Create a gzip compressed string

```
string gzencode (string data [, int level])
```

This function returns a compressed version of the input *data* compatible with the output of the **gzip** program, or false if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be 1.

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Příklad 1. Creating a gzip file

```
<?php
$data = implode("", "bigfile.txt");
$gzdata = gzencode($data, 9);
$fp = fopen("bigfile.txt.gz", "w");
fwrite($fp, $gzdata);
fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1952.txt) (RFC 1952).

See also **gzcompress()**. **gzuncompress()**, **gzdeflate()**, **gzinflate()**.

Část V. PEAR: the PHP Extension and Application Repository

Kapitola 23. About PEAR

PEAR is dedicated to Malin Bakken (http://www.pvv.org/~ssb/malin/bilder/mi/twain001.jpg), born 1999-11-21 (the first PEAR code was written just two hours before she was born).

What is PEAR?

PEAR is a code repository for PHP extensions and PHP library code inspired by TeX's CTAN and Perl's CPAN.

The purpose of PEAR is:

- to provide a consistent means for library code authors to share their code with other developers
- to give the PHP community an infrastructure for sharing code
- to define standards that help developers write portable and reusable code
- to provide tools for code maintenance and distribution

Kapitola 24. PEAR Coding Standards

Indenting

Use an indent of 4 spaces, with no tabs. If you use Emacs to edit PEAR code, you should set indent-tabs-mode to nil. Here is an example mode hook that will set up Emacs according to these guidelines (you will need to ensure that it is called when you are editing php files):

```
(defun php-mode-hook ()
  (setq tab-width 4
        c-basic-offset 4
        c-hanging-comment-ender-p nil
        indent-tabs-mode nil))
```

Here are vim rules for the same thing:

```
  set expandtab
  set shiftwidth=4
  set tabstop=4
```

Control Structures

These include if, for, while, switch, etc. Here is an example if statement, since it is the most complicated of them:

```
if ((condition1) || (condition2)) {
    action1;
} elseif ((condition3) && (condition4)) {
    action2;
} else {
    defaultaction;
}
```

Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls.

You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

For switch statements:

```
switch (condition) {
case 1:
    action1;
    break;

case 2:
    action2;
    break;

default:
    defaultaction;
    break;

}
```

Function Calls

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon. Here’s an example:

```
$var = foo($bar, $baz, $quux);
```

As displayed above, there should be one space on either side of an equals sign used to assign the return value of a function to a variable. In the case of a block of related assignments, more space may be inserted to promote readability:

```
$short         = foo($bar);
$long_variable = foo($baz);
```

Function Definitions

Function declaractions follow the "one true brace" convention:

```
function fooFunction($arg1, $arg2 = ”)
{
    if (condition) {
        statement;
    }
    return $val;
}
```

Arguments with default values go at the end of the argument list. Always attempt to return a meaningful value from a function if one is appropriate. Here is a slightly longer example:

```
function connect(&$dsn, $persistent = false)
{
    if (is_array($dsn)) {
        $dsninfo = &$dsn;
    } else {
        $dsninfo = DB::parseDSN($dsn);
    }

    if (!$dsninfo || !$dsninfo[’phptype’]) {
        return $this->raiseError();
    }

    return true;
}
```

Comments

Inline documentation for classes should follow the PHPDoc convention, similar to Javadoc. More information about PHPDoc can be found here: http://www.phpdoc.de/

Non-documentation comments are strongly encouraged. A general rule of thumb is that if you look at a section of code and think "Wow, I don’t want to try and describe that", you need to comment it before you forget how it works.

C style comments (/* */) and standard C++ comments (//) are both fine. Use of perl/shell style comments (#) is discouraged.

Including Code

Anywhere you are unconditionally including a class file, use **require_once()**. Anywhere you are conditionally including a class file (for example, factory methods), use **include_once()**. Either of these will ensure that class files are included only once. They share the same file list, so you don’t need to worry about mixing them - a file included with **require_once()** will not be included again by **include_once()**.

Poznámka: include_once() and **require_once()** are statements, not functions. You don’t *need* parentheses around the filename to be included.

PHP Code Tags

Always use <?php ?> to delimit PHP code, not the <? ?> shorthand. This is required for PEAR compliance and is also the most portable way to include PHP code on differing operating systems and setups.

Header Comment Blocks

All source code files in the core PEAR distribution should contain the following comment block as the header:

```
/* vim: set expandtab tabstop=4 shiftwidth=4: */
// +----------------------------------------------+
// | PHP version 4.0                                                    |
// +----------------------------------------------+
// | Copyright (c) 1997, 1998, 1999, 2000, 2001 The PHP Group            |
// +----------------------------------------------+
// | This source file is subject to version 2.0 of the PHP license,       |
// | that is bundled with this package in the file LICENSE, and is        |
// | available at through the world-wide-web at                           |
// | http://www.php.net/license/2_02.txt.                                 |
// | If you did not receive a copy of the PHP license and are unable to   |
// | obtain it through the world-wide-web, please send a note to          |
// | license@php.net so we can mail you a copy immediately.               |
// +----------------------------------------------+
// | Authors: Original Author <author@example.com>                        |
// |          Your Name <you@example.com>                                 |
// +----------------------------------------------+
//
// $Id$
```

There’s no hard rule to determine when a new code contributer should be added to the list of authors for a given source file. In general, their changes should fall into the "substantial" category (meaning somewhere around 10% to 20% of code changes). Exceptions could be made for rewriting functions or contributing new logic.

Simple code reorganization or bug fixes would not justify the addition of a new individual to the list of authors.

Files not in the core PEAR repository should have a similar block stating the copyright, the license, and the authors. All files should include the modeline comments to encourage consistency.

CVS Tags

Include the Id CVS vendor tag in each file. As each file is edited, add this tag if it’s not yet present (or replace existing forms such as "Last Modified:", etc.).

Poznámka: We have a custom $Horde tag in Horde cvs to track our versions seperately; we could do the same and make a $PEAR tag, that would remain even if PEAR files were put into another source control system, etc...]

Example URLs

Use "example.com" for all example URLs, per RFC 2606.

Naming Constants

Constants should always be uppercase, with underscores to seperate words. Prefix constant names with the name of the class/package they are used in. For example, the constants used by the `DB::` package all begin with "`DB_`".

LXXXV. PEAR Reference Manual

This chapter contains reference documentation for PEAR components that are distributed with PHP. It is assumed that you are already familiar with objects and classes.

PEAR (unknown)

PEAR base class

Přehled

```
require_once "PEAR.php";
class classname extends PEAR { ... }
```

The PEAR base class provides standard functionality that is used by most PEAR classes. Normally you never make an instance of the PEAR class directly, you use it by subclassing it.

Its key features are:

- request-shutdown object "destructors"
- error handling

PEAR "destructors"

If you inherit PEAR in a class called *ClassName*, you can define a method in it called called *_ClassName* (the class name with an underscore prepended) that will be invoked when the request is over. This is not a destructor in the sense that you can "delete" an object and have the destructor called, but in the sense that PHP gives you a callback in the object when it is done executing. See the example below.

PEAR Error Handling

PEAR's base class also provides a way of passing around more complex errors than a true/false value or a numeric code. A PEAR error is an object that is either an instance of the class PEAR_Error, or some class inheriting PEAR_Error.

One of the design criteria of PEAR's errors is that it should not force a particular type of output on the user, it should be possible to handle errors without any output at all if that is desireable. This makes it possible to handle errors gracefully, also when your output format is different from HTML (for example WML or some other XML format).

The error object can be configured to do a number of things when it is created, such as printing an error message, printing the message and exiting, raising an error with PHP's **trigger_error()** function, invoke a callback, or none of the above. This is typically specified in PEAR_Error's constructor, but all of the parameters are optional, and you can set up defaults for errors generated from each object based on the PEAR class. See the PEAR error examples for how to use it and the PEAR_Error reference for the full details.

The example below shows how to use the PEAR's "poor man's kinda emulated destructors" to implement a simple class that holds the contents of a file, lets you append data to the object and flushes the data back to the file at the end of the request:

Příklad 1. PEAR: emulated destructors

```
require_once "PEAR.php";

class FileContainer extends PEAR
{
    var $file = '';
    var $contents = '';
    var $modified = 0;

    function FileContainer($file)
    {
        $this->PEAR(); // this calls the parent class constructor
        $fp = fopen($file, "r");
        if (!is_resource($fp)) {
            return;
```

```
        }
        while (!empty($data = fread($fp, 2048))) {
            $this->contents .= $data;
     }
        fclose($fp);
    }

    function append($str)
    {
        $this->contents .= $str;
        $this->modified++;
    }

    // The "destructor" is named like the constructor
    // but with an underscore in front.
    function _FileContainer()
    {
        if ($this->modified) {
            $fp = fopen($this->file, "w");
            if (!is_resource($fp)) {
                return;
            }
            fwrite($fp, $this->contents);
            fclose($fp);
        }
    }
}

$fileobj = new FileContainer("testfile");
$fileobj->append("this ends up at the end of the file\n");

// When the request is done and PHP shuts down, $fileobj's
// "destructor" is called and updates the file on disk.
```

Poznámka: PEAR "destructors" use PHP's shutdown callbacks (**register_shutdown_function()**), and you can't output anything from these when PHP is running in a web server. So anything printed in a "destructor" gets lost except when PHP is used in command-line mode. Bummer.

The next examples illustrate different ways of using PEAR's error handling mechanism.

Příklad 2. PEAR error example (1)

```
function mysockopen($host = "localhost", $port = 8090)
{
    $fp = fsockopen($host, $port, $errno, $errstr);
    if (!is_resource($fp)) {
        return new PEAR_Error($errstr, $errno);
    }
    return $fp;
}

$sock = mysockopen();
if (PEAR::isError($sock)) {
    print "mysockopen error: ".$sock->getMessage()."<BR>\n"
}
```

This example shows a wrapper to **fsockopen()** that delivers the error code and message (if any) returned by fsockopen in a PEAR error object. Notice that **PEAR::isError()** is used to detect whether a value is a PEAR error.

PEAR_Error's mode of operation in this example is simply returning the error object and leaving the rest to the user (programmer). This is the default error mode.

In the next example we're showing how to use default error modes:

Příklad 3. PEAR error example (2)

```
class TCP_Socket extends PEAR
{
    var $sock;

    function TCP_Socket()
    {
        $this->PEAR();
    }

    function connect($host, $port)
    {
        $sock = fsockopen($host, $port, $errno, $errstr);
        if (!is_resource($sock)) {
            return $this->raiseError($errstr, $errno);
        }
    }
}

$sock = new TCP_Socket;
$sock->setErrorHandling(PEAR_ERROR_DIE);
$sock->connect("localhost", 8090);
print "still alive<BR>\n";
```

Here, we set the default error mode to `PEAR_ERROR_DIE`, and since we don't specify any error mode in the raiseError call (that'd be the third parameter), raiseError uses the default error mode and exits if fsockopen fails.

PEAR_Error (unknown)

PEAR error mechanism base class

Přehled

```
$err = new PEAR_Error($msg);
```

An error object has a mode of operation that can be set with one of the following constants:

PEAR_ERROR_RETURN

Just return the object, don't do anything special in PEAR_Error's constructor.

PEAR_ERROR_PRINT

Print the error message in the constructor. The execution is not interrupted.

PEAR_ERROR_TRIGGER

Use PHP's **trigger_error()** function to raise an internal error in PHP. The execution is aborted if you have defined your own PHP error handler or if you set the error severity to E_USER_ERROR.

PEAR_ERROR_DIE

Print the error message and exit. Execution is of course aborted.

PEAR_ERROR_CALLBACK

Use a callback function or method to handle errors. Execution is aborted.

PEAR_Error::PEAR_Error ([*message code mode options userinfo*]]]]])

Description

PEAR_Error constructor. Parameters:

message

error message, defaults to "unknown error"

code

error code (optional)

mode

Mode of operation. See the error modes section for details.

options

If the mode of can have any options specified, use this parameter. Currently the "trigger" and "callback" modes are the only using the options parameter. For trigger mode, this parameter is one of `E_USER_NOTICE`, `E_USER_WARNING` or `E_USER_ERROR`. For callback mode, this parameter should contain either the callback function name (string), or a two-element (object, string) array representing an object and a method name.

Část VI. Dodatky

Dodatek A. Migrating from older versions of PHP

Migrating from PHP 3 to PHP 4

Migration from PHP 3 to PHP 4 is relatively easy, and should not require you to change your code in any way. There are minor incompatibilities between the two versions; You may want to check the incompatibilities list to make sure that you're indeed not affected by them (the chances you're affected by these incompatibilities are extremely slim).

Running PHP 3 and PHP 4 concurrently

Recent operating systems provide the ability to perform versioning and scoping. This features make it possible to let PHP 3 and PHP 4 run as concurrent modules in one Apache server.

This feature is known to work on the following platforms:

- Linux with recent binutils (binutils 2.9.1.0.25 tested)
- Solaris 2.5 or better
- FreeBSD (3.2, 4.0 tested)

To enable it, configure PHP3 and PHP4 to use APXS (–with-apxs) and the necessary link extensions (–enable-versioning). Otherwise, all standard installations instructions apply. For example:

```
$ ./configure \
  -with-apxs=/apache/bin/apxs \
  -enable-versioning \
  -with-mysql \
  -enable-track-vars
```

Migrating Configuration Files

Global Configuration File

The global configuration file, php3.ini, has changed its name to php.ini.

Apache Configuration Files

The MIME types recognized by the PHP module have changed.

```
application/x-httpd-php3        ->    application/x-httpd-php
application/x-httpd-php3-source ->    application/x-httpd-php-source
```

You can make your configuration files work with both versions of PHP (depending on which one is currently compiled into the server), using the following syntax:

```
AddType  application/x-httpd-php3        .php3
AddType  application/x-httpd-php3-source .php3s

AddType  application/x-httpd-php         .php
AddType  application/x-httpd-php-source  .phps
```

In addition, the PHP directive names for Apache have changed.

Starting with PHP 4.0, there are only four Apache directives that relate to PHP:

```
php_value [PHP directive name] [value]
php_flag [PHP directive name] [On|Off]
php_admin_value [PHP directive name] [value]
php_admin_flag [PHP directive name] [On|Off]
```

There are two differences between the Admin values and the non admin values:

- Admin values (or flags) can only appear in the server-wide apache configuration files (e.g., httpd.conf).
- Standard values (or flags) cannot control certain PHP directives, for example - safe mode (if you could override safe mode settings in .htaccess files, it would defeat safe-mode's purpose). In contrast, Admin values can modify the value of any PHP directive.

To make the transition process easier, PHP 4.0 is bundled with scripts that automatically convert your Apache configuration and .htaccess files to work with both PHP 3.0 and PHP 4.0. These scripts do NOT convert the mime type lines! You have to convert these yourself.

To convert your Apache configuration files, run the apconf-conv.sh script (available in the scripts/apache/ directory). For example:

```
~/php4/scripts/apache:#  ./apconf-conv.sh /usr/local/apache/conf/httpd.conf
```

Your original configuration file will be saved in httpd.conf.orig.

To convert your .htaccess files, run the aphtaccess-conv.sh script (available in the scripts/apache/ directory as well):

```
~/php4/scripts/apache:#  find / -name .htaccess -exec ./aphtaccess-conv.sh {} \;
```

Likewise, your old .htaccess files will be saved with an .orig prefix.

The conversion scripts require awk to be installed.

Migrating from PHP/FI 2.0 to PHP 3.0

About the incompatibilities in 3.0

PHP 3.0 is rewritten from the ground up. It has a proper parser that is much more robust and consistent than 2.0's. 3.0 is also significantly faster, and uses less memory. However, some of these improvements have not been possible without compatibility changes, both in syntax and functionality.

In addition, PHP's developers have tried to clean up both PHP's syntax and semantics in version 3.0, and this has also caused some incompatibilities. In the long run, we believe that these changes are for the better.

This chapter will try to guide you through the incompatibilities you might run into when going from PHP/FI 2.0 to PHP 3.0 and help you resolve them. New features are not mentioned here unless necessary.

A conversion program that can automatically convert your old PHP/FI 2.0 scripts exists. It can be found in the `convertor` subdirectory of the PHP 3.0 distribution. This program only catches the syntax changes though, so you should read this chapter carefully anyway.

Start/end tags

The first thing you probably will notice is that PHP's start and end tags have changed. The old `<? >` form has been replaced by three new possible forms:

Příklad A-1. Migration: old start/end tags

```
<? echo "This is PHP/FI 2.0 code.\n"; >
```

As of version 2.0, PHP/FI also supports this variation:

Příklad A-2. Migration: first new start/end tags

```
<? echo "This is PHP 3.0 code!\n"; ?>
```

Notice that the end tag now consists of a question mark and a greater-than character instead of just greater-than. However, if you plan on using XML on your server, you will get problems with the first new variant, because PHP may try to execute the XML markup in XML documents as PHP code. Because of this, the following variation was introduced:

Příklad A-3. Migration: second new start/end tags

```
<?php echo "This is PHP 3.0 code!\n"; ?>
```

Some people have had problems with editors that don't understand the processing instruction tags at all. Microsoft FrontPage is one such editor, and as a workaround for these, the following variation was introduced as well:

Příklad A-4. Migration: third new start/end tags

```
<script language="php">

  echo "This is PHP 3.0 code!\n";

</script>
```

if..endif syntax

The 'alternative' way to write if/elseif/else statements, using if(); elseif(); else; endif; cannot be efficiently implemented without adding a large amount of complexity to the 3.0 parser. Because of this, the syntax has been changed:

Příklad A-5. Migration: old if..endif syntax

```
if ($foo);
    echo "yep\n";
elseif ($bar);
    echo "almost\n";
else;
    echo "nope\n";
endif;
```

Příklad A-6. Migration: new if..endif syntax

```
if ($foo):
    echo "yep\n";
elseif ($bar):
    echo "almost\n";
else:
    echo "nope\n";
endif;
```

Notice that the semicolons have been replaced by colons in all statements but the one terminating the expression (endif).

while syntax

Just like with if..endif, the syntax of while..endwhile has changed as well:

Příklad A-7. Migration: old while..endwhile syntax

```
while ($more_to_come);
    ...
endwhile;
```

Příklad A-8. Migration: new while..endwhile syntax

```
while ($more_to_come):
    ...
endwhile;
```

> **Varování**
>
> If you use the old while..endwhile syntax in PHP 3.0, you will get a never-ending loop.

Expression types

PHP/FI 2.0 used the left side of expressions to determine what type the result should be. PHP 3.0 takes both sides into account when determining result types, and this may cause 2.0 scripts to behave unexpectedly in 3.0.

Consider this example:

```
$a[0]=5;
$a[1]=7;

$key = key($a);
while ("" != $key) {
    echo "$keyn";
    next($a);
}
```

In PHP/FI 2.0, this would display both of $a's indices. In PHP 3.0, it wouldn't display anything. The reason is that in PHP 2.0, because the left argument's type was string, a string comparison was made, and indeed `""` does not equal `"0"`, and the loop went through. In PHP 3.0, when a string is compared with an integer, an integer comparison is made (the string is converted to an integer). This results in comparing `atoi("")` which is `0`, and `variablelist` which is also `0`, and since `0==0`, the loop doesn't go through even once.

The fix for this is simple. Replace the while statement with:

```
while ((string)$key != "") {
```

Error messages have changed

PHP 3.0's error messages are usually more accurate than 2.0's were, but you no longer get to see the code fragment causing the error. You will be supplied with a file name and a line number for the error, though.

Short-circuited boolean evaluation

In PHP 3.0 boolean evaluation is short-circuited. This means that in an expression like `(1 || test_me())`, the function **test_me()** would not be executed since nothing can change the result of the expression after the `1`.

This is a minor compatibility issue, but may cause unexpected side-effects.

Function true/false return values

Most internal functions have been rewritten so they return TRUE when successful and FALSE when failing, as opposed to 0 and -1 in PHP/FI 2.0, respectively. The new behaviour allows for more logical code, like `$fp = fopen("/your/file") or fail("darn!");`. Because PHP/FI 2.0 had no clear rules for what functions should return when they failed, most such scripts will probably have to be checked manually after using the 2.0 to 3.0 convertor.

Příklad A-9. Migration from 2.0: return values, old code

```
$fp = fopen($file, "r");
if ($fp == -1);
    echo("Could not open $file for reading<br>\n");
endif;
```

Příklad A-10. Migration from 2.0: return values, new code

```
$fp = @fopen($file, "r") or print("Could not open $file for reading<br>\n");
```

Other incompatibilities

- The PHP 3.0 Apache module no longer supports Apache versions prior to 1.2. Apache 1.2 or later is required.
- **echo()** no longer supports a format string. Use the **printf()** function instead.
- In PHP/FI 2.0, an implementation side-effect caused `$foo[0]` to have the same effect as `$foo`. This is not true for PHP 3.0.
- Reading arrays with `$array[]` is no longer supported

  That is, you cannot traverse an array by having a loop that does `$data = $array[]`. Use **current()** and **next()** instead.

  Also, `$array1[] = $array2` does not append the values of `$array2` to `$array1`, but appends `$array2` as the last entry of `$array1`. See also multidimensional array support.

- `"+"` is no longer overloaded as a concatenation operator for strings, instead it converts it's arguments to numbers and performs numeric addition. Use `"."` instead.

Příklad A-11. Migration from 2.0: concatenation for strings

```
echo "1" + "1";
```

In PHP 2.0 this would echo 11, in PHP 3.0 it would echo 2. Instead use:

```
echo "1"."1";
$a = 1;
$b = 1;
echo $a + $b;
```

This would echo 2 in both PHP 2.0 and 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

This will echo 11 in PHP 3.0.

Dodatek B. Migrating from PHP 3.0 to PHP 4.0

What has changed in PHP 4.0

PHP 4.0 and the integrated Zend engine have greatly inproved PHPs performance and capabilities, but great care has been taken to break as little existing code as possible. So migrating your code from PHP 3.0 to 4.0 should be much easier than migrating from PHP/FI 2.0 to PHP 3.0. A lot of existing PHP 3.0 code should be ready to run without changes, but you should still know about the few differences and take care to test your code before switching versions in production environments. The following should give you some hints about what to look for.

Parser behavior

Parsing and execution are now two completely seperated steps, no execution of a files code will happen until the complete file and everything it requires has completely and successfully been parsed.

One of the new requirenments introduced with this split is that required and included files now have to be syntactically complete. You can no longer spread the different controling parts of a control structure across file boundaries. That is you cannot start a `for` or `while` loop, an `if` statement or a `switch` block in one file and have the end of loop, `else`, `endif`, `case` or `break` statements in a different file.

It still perfectly legal to include additional code within loops or other control structures, only the controling keywords and corresponding curly braces `{...}` have to be within the same compile unit (file or **eval()**ed string).

This should not harm to much as spreading code like this should be considered as very bad style anyway.

Another thing no longer possible, though rarely seen in PHP 3.0 code is returning values from a required file. Returning a value from an included file is still possible.

Error reporting

Configuration changes

With PHP 3.0 the error reporting level was set as a simple numeric value formed by summing up the numbers related to different error levels. Usual values where 15 for reporting all errors and warnings or 7 for reporting everything but simple notice messages reporting bad style and things like that.

PHP 4.0 now has a greater set of different error and warning levels and comes with a configuration parser that now allows for symbolic constants to be used for setting up the intended behavior.

Error reporting level should now be configured by explicitly taking away the warning levels you do not want to generate error messages by x-oring them from the symbolic constant `E_ALL`. Sounds complicated? Well, lets say you want the error reporting system to report all but the simple style warnings that are categorized by the symbolic constant `E_NOTICE`. Then you'll put the following into your `php.ini`: `error_reporting = E_ALL & ~ ( E_NOTICE )`. If you want to suppress warnings too you add up the appropriate constant within the braces using the binary or operator '|': `error_reporting= E_ALL & ~ ( E_NOTICE | E_WARNING )`.

Varování

Using the old values 7 and 15 for setting up error reporting is a very bad idea as this suppresses some of the newly added error classes including parese errors. This may lead to very strange behavior as scripts might no longer work without error messages showing up anywhere.

This has lead to a lot of unreproduceable bug reports in the past where people reported script engine problems they were not capable to track down while the true case was usually some missing '}' in a required file that the parser was not able to report due to a misconfigured error reporting system.

So checking your error reporting setup should be the first thing to do whenever your scripts silently die. The Zend enginge can be considererd mature enough nowadays to not cause this kind of strange behavior.

Additional warning messages

A lot of existing PHP 3.0 code uses language constructs that should be considered as very bad style as this code, while doing the intended thing now, could easily be broken by changes in other places. PHP 4.0 will output a lot of notice messages in such situations where PHP 3.0 didn't. The easy fix is to just turn off E_NOTICE messages, but it is usually a good idea to fix the code instead.

The most common case that will now produce notice messages is the use of unquoted string constants as array indices. Both PHP 3.0 and 4.0 will fall back to interprete theese as strings if no keyword or constant is known by that name, but whenever a constant by that name had been defined anywhere else in the code it might break your script. This can even become a security risk if some intruder manages to redefine string constants in a way that makes your script give him access rights he wasn't intended to have. So PHP 4.0 will now warn you whenever you use unquoted string constants as for example in `$HTTP_SERVER_VARS[REQUEST_METHOD]`. Changing it to `$HTTP_SERVER_VARS['REQUEST_METHOD']` will make the parser happy and greatly improve the style and security of your code.

Another thing PHP 4.0 will now tell you about is the use of uninitialized variables or array elements.

Initializers

Static variable and class member initializers only accept scalar values while in PHP 3.0 they accepted any valid expression. This is, once again, due to the split between parsing and execution as no code has yet been executed when the parser sees the initializer.

For classes you should use constructors to initialize member variables instead. For static variables anything but a simple static value rarely makes sense anyway.

`empty("0")`

The perhaps most cotroversal change in behavior has happend to the behavior of the **empty()**. A String containing only the character '0' (zero) is now considered empty while it wasn't in PHP 3.0.

This new behavior makes sense in web applications, with all input fields returning strings even if numeric input is requested, and with PHP's capabilities of automatic type conversion. But on the other had it might break your code in a rather subtele was, leading to misbehavior that is hard to track down if you do not know about what to look for.

Missing functions

While PHP 4.0 comes with a lot of new features, functions and extensions, you may still find some functions from version 3.0 missing. A small number of core functions has vanished because they do not work with the new scheme of splitting parsing and execution as introduced into 4.0 with the Zend engine. Other functions and even complete extensions have become obsolete as newer functions and extensions serve the same task better and/or in a more general way. Some functions just simply haven't been ported yet and finally some functions or extensions may be missing due to license conflicts.

Functions missing due to conceptual changes

As PHP 4.0 now seperates parsing from execution it is no longer possible to change the behavior of the parser (now embedded in the Zend engine) at runtime as parsing already happend by then. So the function **short_tags()** has ceased to exist. You can still change the parsers behavior by setting appropriate values in the `php.ini` file.

Another feature from PHP 3.0 that didn't make it into 4.0 is the experimental debugging interface as described in ??? in this manual. A seperate true debuger is promissed to be released as a Zend product but hasn't become visible yet.

Deprecate functions and extensions

The Adabas and Solid database extensions are no more. Long live the unified ODBC extension instead.

Changed status for unset()

unset(), although still available, is implemented a literal different in PHP 4.0 so that it no longer counts as a 'real' function.

This has no direct consequences as the behavior of **unset()** didn't change, but testing for "unset" usign **function_exists()** will return `false` as it would with othern lowlevel functions like **echo()**.

Another more practical change is that it is no longer possible to call **unset()** indirectly, that is `$func="unset"; $func($somevar)` won't work anymore.

PHP 3.0 extension

Extensions written for PHP 3.0 will not work with PHP 4.0 anymore, neither as binaries nor at the source level. It is not to difficult to port your extensions to PHP 4.0 if you have access to the original sources. A detailed description of the actual porting process is not part of this text (yet).

Variable substitution in strings

PHP 4.0 adds a new mechanism to variable substitution in strings. You can now finally access object member variables and elements from multidimensional arrays within strings.

To do so you have to enclose your variables with curly braces with the dollar sign immediately following the opening brace: `{$...}`

To embed the value of an object member variable into a string you simply write `"text {$obj->member} text"` while in PHP 3.0 you had to use something like `"text ".$obj->member." text"`.

This should lead to more readable code, while it may break existing scripts written for PHP 3.0. But you can easily check for this kind of problem by checking for the character combination `{$` in your code and by replacing it with `\{$` with your favourite search-and-replace tool.

Cookies

PHP 3.0 hat the bad habit of setting cookies in the reverse order of the **setcookie()** calls in your code. PHP 4.0 breaks with this habbit and creates the cookie header lines in exactly the same order as you set the cookies in the code.

This might break some existing code, but the old behaviour was so strange to understand that it deserved a change to prevent further problems in the future.

Dodatek C. PHP development

Adding functions to PHP 3

Function Prototype

All functions look like this:

```
void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {

}
```

Even if your function doesn't take any arguments, this is how it is called.

Function Arguments

Arguments are always of type pval. This type contains a union which has the actual type of the argument. So, if your function takes two arguments, you would do something like the following at the top of your function:

Příklad C-1. Fetching function arguments

```
pval *arg1, *arg2;
if (ARG_COUNT(ht) != 2 || getParameters(ht,2,&arg1,&arg2)==FAILURE) {
   WRONG_PARAM_COUNT;
}
```

NOTE: Arguments can be passed either by value or by reference. In both cases you will need to pass &(pval *) to getParameters. If you want to check if the n'th parameter was sent to you by reference or not, you can use the function, ParameterPassedByReference(ht,n). It will return either 1 or 0.

When you change any of the passed parameters, whether they are sent by reference or by value, you can either start over with the parameter by calling pval_destructor on it, or if it's an ARRAY you want to add to, you can use functions similar to the ones in internal_functions.h which manipulate return_value as an ARRAY.

Also if you change a parameter to IS_STRING make sure you first assign the new estrdup()'ed string and the string length, and only later change the type to IS_STRING. If you change the string of a parameter which already IS_STRING or IS_ARRAY you should run pval_destructor on it first.

Variable Function Arguments

A function can take a variable number of arguments. If your function can take either 2 or 3 arguments, use the following:

Příklad C-2. Variable function arguments

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);

if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Using the Function Arguments

The type of each argument is stored in the pval type field. This type can be any of the following:

Tabulka C-1. PHP Internal Types

IS_STRING	String
IS_DOUBLE	Double-precision floating point
IS_LONG	Long integer
IS_ARRAY	Array
IS_EMPTY	None
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (if some of these cannot be passed to a function - delete)
IS_CLASS	??
IS_OBJECT	??

If you get an argument of one type and would like to use it as another, or if you just want to force the argument to be of a certain type, you can use one of the following conversion functions:

```
convert_to_long(arg1);
convert_to_double(arg1);
convert_to_string(arg1);
convert_to_boolean_long(arg1); /* If the string is "" or "0" it becomes 0, 1 other-
wise */
convert_string_to_number(arg1);  /* Converts string to either LONG or DOUBLE depend-
ing on string */
```

These function all do in-place conversion. They do not return anything.

The actual argument is stored in a union; the members are:

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Memory Management in Functions

Any memory needed by a function should be allocated with either emalloc() or estrdup(). These are memory handling abstraction functions that look and smell like the normal malloc() and strdup() functions. Memory should be freed with efree().

There are two kinds of memory in this program: memory which is returned to the parser in a variable, and memory which you need for temporary storage in your internal function. When you assign a string to a variable which is returned to the parser you need to make sure you first allocate the memory with either emalloc() or estrdup(). This memory should NEVER be freed by you, unless you later in the same function overwrite your original assignment (this kind of programming practice is not good though).

For any temporary/permanent memory you need in your functions/library you should use the three emalloc(), estrdup(), and efree() functions. They behave EXACTLY like their counterpart functions. Anything you emalloc() or estrdup() you have to efree() at some point or another, unless it’s supposed to stick around until the end of the program; otherwise, there will be a memory leak. The meaning of "the functions behave exactly like their counterparts" is: if you efree() something which was not emalloc()’ed nor estrdup()’ed you might get a segmentation fault. So please take care and free all of your wasted memory.

If you compile with "-DDEBUG", PHP 3 will print out a list of all memory that was allocated using emalloc() and estrdup() but never freed with efree() when it is done running the specified script.

Setting Variables in the Symbol Table

A number of macros are available which make it easier to set a variable in the symbol table:

- SET_VAR_STRING(name,value) [1]
- SET_VAR_DOUBLE(name,value)
- SET_VAR_LONG(name,value)

[1]

Symbol tables in PHP 3.0 are implemented as hash tables. At any given time, &symbol_table is a pointer to the 'main' symbol table, and active_symbol_table points to the currently active symbol table (these may be identical like in startup, or different, if you're inside a function).

The following examples use 'active_symbol_table'. You should replace it with &symbol_table if you specifically want to work with the 'main' symbol table. Also, the same functions may be applied to arrays, as explained below.

Příklad C-3. Checking whether $foo exists in a symbol table

```
if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) { exists... }
else { doesn't exist }
```

Příklad C-4. Finding a variable's size in a symbol table

```
hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
check(pvalue.type);
```

Arrays in PHP 3.0 are implemented using the same hashtables as symbol tables. This means the two above functions can also be used to check variables inside arrays.

If you want to define a new array in a symbol table, you should do the following.

First, you may want to check whether it exists and abort appropiately, using hash_exists() or hash_find().

Next, initialize the array:

Příklad C-5. Initializing a new array

```
pval arr;

if (array_init(&arr) == FAILURE) { failed... };
hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
```

This code declares a new array, named $foo, in the active symbol table. This array is empty.

Here's how to add new entries to it:

Příklad C-6. Adding entries to a new array

```
pval entry;

entry.type = IS_LONG;
entry.value.lval = 5;

/* defines $foo["bar"] = 5 */
hash_update(arr.value.ht,"bar",sizeof("bar"),&entry,sizeof(pval),NULL);

/* defines $foo[7] = 5 */
hash_index_update(arr.value.ht,7,&entry,sizeof(pval),NULL);

/* defines the next free place in $foo[],
 * $foo[8], to be 5 (works like php2)
```

```
 */
hash_next_index_insert(arr.value.ht,&entry,sizeof(pval),NULL);
```

If you’d like to modify a value that you inserted to a hash, you must first retrieve it from the hash. To prevent that overhead, you can supply a pval ** to the hash add function, and it’ll be updated with the pval * address of the inserted element inside the hash. If that value is NULL (like in all of the above examples) - that parameter is ignored.

hash_next_index_insert() uses more or less the same logic as "$foo[] = bar;" in PHP 2.0.

If you are building an array to return from a function, you can initialize the array just like above by doing:

```
if (array_init(return_value) == FAILURE) { failed...; }
```

...and then adding values with the helper functions:

```
add_next_index_long(return_value,long_value);
add_next_index_double(return_value,double_value);
add_next_index_string(return_value,estrdup(string_value));
```

Of course, if the adding isn’t done right after the array initialization, you’d probably have to look for the array first:

```
pval *arr;

if (hash_find(active_symbol_table,"foo",sizeof("foo"),(void **)&arr)==FAILURE) { can’t find...
else { use arr->value.ht... }
```

Note that hash_find receives a pointer to a pval pointer, and not a pval pointer.

Just about any hash function returns SUCCESS or FAILURE (except for hash_exists(), which returns a boolean truth value).

Returning simple values

A number of macros are available to make returning values from a function easier.

The RETURN_* macros all set the return value and return from the function:

- RETURN
- RETURN_FALSE
- RETURN_TRUE
- RETURN_LONG(l)
- RETURN_STRING(s,dup) If dup is true, duplicates the string
- RETURN_STRINGL(s,l,dup) Return string (s) specifying length (l).
- RETURN_DOUBLE(d)

The RETVAL_* macros set the return value, but do not return.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) If dup is true, duplicates the string
- RETVAL_STRINGL(s,l,dup) Return string (s) specifying length (l).
- RETVAL_DOUBLE(d)

The string macros above will all estrdup() the passed ’s’ argument, so you can safely free the argument after calling the macro, or alternatively use statically allocated memory.

If your function returns boolean success/error responses, always use RETURN_TRUE and RETURN_FALSE respectively.

Returning complex values

Your function can also return a complex data type such as an object or an array.

Returning an object:

1. Call object_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.
3. Possibly, register functions for this object. In order to obtain values from the object, the function would have to fetch "this" from the active_symbol_table. Its type should be IS_OBJECT, and it’s basically a regular hash table (i.e., you can use regular hash functions on .value.ht). The actual registration of the function can be done using:

   ```
   add_method( return_value, function_name, function_ptr );
   ```

The functions used to populate an object are:

- add_property_long(return_value, property_name, l) - Add a property named ’property_name’, of type long, equal to ’l’
- add_property_double(return_value, property_name, d) - Same, only adds a double
- add_property_string(return_value, property_name, str) - Same, only adds a string
- add_property_stringl(return_value, property_name, str, l) - Same, only adds a string of length ’l’

Returning an array:

1. Call array_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.

The functions used to populate an array are:

- add_assoc_long(return_value,key,l) - add associative entry with key ’key’ and long value ’l’
- add_assoc_double(return_value,key,d)
- add_assoc_string(return_value,key,str,duplicate)
- add_assoc_stringl(return_value,key,str,length,duplicate) specify the string length
- add_index_long(return_value,index,l) - add entry in index ’index’ with long value ’l’
- add_index_double(return_value,index,d)
- add_index_string(return_value,index,str)
- add_index_stringl(return_value,index,str,length) - specify the string length
- add_next_index_long(return_value,l) - add an array entry in the next free offset with long value ’l’
- add_next_index_double(return_value,d)
- add_next_index_string(return_value,str)
- add_next_index_stringl(return_value,str,length) - specify the string length

Using the resource list

PHP 3.0 has a standard way of dealing with various types of resources. This replaces all of the local linked lists in PHP 2.0.

Available functions:

- php3_list_insert(ptr, type) - returns the 'id' of the newly inserted resource
- php3_list_delete(id) - delete the resource with the specified id
- php3_list_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

Typically, these functions are used for SQL drivers but they can be used for anything else; for instance, maintaining file descriptors.

Typical list code would look like this:

Příklad C-7. Adding a new resource

```
RESOURCE *resource;

/* ...allocate memory for resource and acquire resource... */
/* add a new resource to the list */
return_value->value.lval = php3_list_insert((void *) resource, LE_RESOURCE_TYPE);
return_value->type = IS_LONG;
```

Příklad C-8. Using an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;

convert_to_long(resource_id);
resource = php3_list_find(resource_id->value.lval, &type);
if (type != LE_RESOURCE_TYPE) {
php3_error(E_WARNING,"resource index %d has the wrong type",resource_id->value.lval);
RETURN_FALSE;
}
/* ...use resource... */
```

Příklad C-9. Deleting an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;

convert_to_long(resource_id);
php3_list_delete(resource_id->value.lval);
```

The resource types should be registered in php3_list.h, in enum list_entry_type. In addition, one should add shutdown code for any new resource type defined, in list.c's list_entry_destructor() (even if you don't have anything to do on shutdown, you must add an empty case).

Using the persistent resource table

PHP 3.0 has a standard way of storing persistent resources (i.e., resources that are kept in between hits). The first module to use this feature was the MySQL module, and mSQL followed it, so one can get the general impression of how a persistent resource should be used by reading mysql.c. The functions you should look at are:

php3_mysql_do_connect

php3_mysql_connect()
php3_mysql_pconnect()

The general idea of persistence modules is this:

1. Code all of your module to work with the regular resource list mentioned in section (9).
2. Code extra connect functions that check if the resource already exists in the persistent resource list. If it does, register it as in the regular resource list as a pointer to the persistent resource list (because of 1., the rest of the code should work immediately). If it doesn't, then create it, add it to the persistent resource list AND add a pointer to it from the regular resource list, so all of the code would work since it's in the regular resource list, but on the next connect, the resource would be found in the persistent resource list and be used without having to recreate it. You should register these resources with a different type (e.g. LE_MYSQL_LINK for non-persistent link and LE_MYSQL_PLINK for a persistent link).

If you read mysql.c, you'll notice that except for the more complex connect function, nothing in the rest of the module has to be changed.

The very same interface exists for the regular resource list and the persistent resource list, only 'list' is replaced with 'plist':

- php3_plist_insert(ptr, type) - returns the 'id' of the newly inserted resource
- php3_plist_delete(id) - delete the resource with the specified id
- php3_plist_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

However, it's more than likely that these functions would prove to be useless for you when trying to implement a persistent module. Typically, one would want to use the fact that the persistent resource list is really a hash table. For instance, in the MySQL/mSQL modules, when there's a pconnect() call (persistent connect), the function builds a string out of the host/user/passwd that were passed to the function, and hashes the SQL link with this string as a key. The next time someone calls a pconnect() with the same host/user/passwd, the same key would be generated, and the function would find the SQL link in the persistent list.

Until further documented, you should look at mysql.c or msql.c to see how one should use the plist's hash table abilities.

One important thing to note: resources going into the persistent resource list must *NOT* be allocated with PHP's memory manager, i.e., they should NOT be created with emalloc(), estrdup(), etc. Rather, one should use the regular malloc(), strdup(), etc. The reason for this is simple - at the end of the request (end of the hit), every memory chunk that was allocated using PHP's memory manager is deleted. Since the persistent list isn't supposed to be erased at the end of a request, one mustn't use PHP's memory manager for allocating resources that go to it.

When you register a resource that's going to be in the persistent list, you should add destructors to it both in the non-persistent list and in the persistent list. The destructor in the non-persistent list destructor shouldn't do anything. The one in the persistent list destructor should properly free any resources obtained by that type (e.g. memory, SQL links, etc). Just like with the non-persistent resources, you *MUST* add destructors for every resource, even it requires no destructotion and the destructor would be empty. Remember, since emalloc() and friends aren't to be used in conjunction with the persistent list, you mustn't use efree() here either.

Adding runtime configuration directives

Many of the features of PHP 3 can be configured at runtime. These configuration directives can appear in either the designated php3.ini file, or in the case of the Apache module version in the Apache .conf files. The advantage of having them in the Apache .conf files is that they can be configured on a per-directory basis. This means that one directory may have a certain safemodeexecdir for example, while another directory may have another. This configuration granularity is especially handy when a server supports multiple virtual hosts.

The steps required to add a new directive:

1. Add directive to php3_ini_structure struct in mod_php3.h.

2. In main.c, edit the php3_module_startup function and add the appropriate cfg_get_string() or cfg_get_long() call.
3. Add the directive, restrictions and a comment to the php3_commands structure in mod_php3.c. Note the restrictions part. RSRC_CONF are directives that can only be present in the actual Apache .conf files. Any OR_OPTIONS directives can be present anywhere, include normal .htaccess files.
4. In either php3take1handler() or php3flaghandler() add the appropriate entry for your directive.
5. In the configuration section of the _php3_info() function in functions/info.c you need to add your new directive.
6. And last, you of course have to use your new directive somewhere. It will be addressable as php3_ini.directive.

Calling User Functions

To call user functions from an internal function, you should use the **call_user_function()** function.

call_user_function() returns SUCCESS on success, and FAILURE in case the function cannot be found. You should check that return value! If it returns SUCCESS, you are responsible for destroying the retval pval yourself (or return it as the return value of your function). If it returns FAILURE, the value of retval is undefined, and you mustn't touch it.

All internal functions that call user functions *must* be reentrant. Among other things, this means they must not use globals or static variables.

call_user_function() takes six arguments:

HashTable *function_table

This is the hash table in which the function is to be looked up.

pval *object

This is a pointer to an object on which the function is invoked. This should be NULL if a global function is called. If it's not NULL (i.e. it points to an object), the function_table argument is ignored, and instead taken from the object's hash. The object *may* be modified by the function that is invoked on it (that function will have access to it via $this). If for some reason you don't want that to happen, send a copy of the object instead.

pval *function_name

The name of the function to call. Must be a pval of type IS_STRING with function_name.str.val and function_name.str.len set to the appropriate values. The function_name is modified by call_user_function() - it's converted to lowercase. If you need to preserve the case, send a copy of the function name instead.

pval *retval

A pointer to a pval structure, into which the return value of the invoked function is saved. The structure must be previously allocated - **call_user_function()** does NOT allocate it by itself.

int param_count

The number of parameters being passed to the function.

pval *params[]

An array of pointers to values that will be passed as arguments to the function, the first argument being in offset 0, the second in offset 1, etc. The array is an array of pointers to pval's; The pointers are sent as-is to the function, which means if the function modifies its arguments, the original values are changed (passing by reference). If you don't want that behavior, pass a copy instead.

Reporting Errors

To report errors from an internal function, you should call the **php3_error()** function. This takes at least two parameters – the first is the level of the error, the second is the format string for the error message (as in a standard **printf()** call), and any following arguments are the parameters for the format string. The error levels are:

E_NOTICE

Notices are not printed by default, and indicate that the script encountered something that could indicate an error, but could also happen in the normal course of running a script. For example, trying to access the value of a variable which has not been set, or calling **stat()** on a file that doesn't exist.

E_WARNING

Warnings are printed by default, but do not interrupt script execution. These indicate a problem that should have been trapped by the script before the call was made. For example, calling **ereg()** with an invalid regular expression.

E_ERROR

Errors are also printed by default, and execution of the script is halted after the function returns. These indicate errors that can not be recovered from, such as a memory allocation problem.

E_PARSE

Parse errors should only be generated by the parser. The code is listed here only for the sake of completeness.

E_CORE_ERROR

This is like an E_ERROR, except it is generated by the core of PHP. Functions should not generate this type of error.

E_CORE_WARNING

This is like an E_WARNING, except it is generated by the core of PHP. Functions should not generate this type of error.

E_COMPILE_ERROR

This is like an E_ERROR, except it is generated by the Zend Scripting Engine. Functions should not generate this type of error.

E_COMPILE_WARNING

This is like an E_WARNING, except it is generated by the Zend Scripting Engine. Functions should not generate this type of error.

E_USER_ERROR

This is like an E_ERROR, except it is generated in PHP code by using the PHP function **trigger_error()**. Functions should not generate this type of error.

E_USER_WARNING

This is like an E_WARNING, except it is generated by using the PHP function **trigger_error()**. Functions should not generate this type of error.

E_USER_NOTICE

This is like an E_NOTICE, except it is generated by using the PHP function **trigger_error()**. Functions should not generate this type of error.

Poznámky

Be careful here. The value part must be malloc’ed manually because the memory management code will try to free this pointer later. Do not pass statically allocated memory into a SET_VAR_STRING.

Dodatek D. The PHP Debugger

About the debugger

PHP 3 includes support for a network-based debugger.

PHP 4 does not yet have a similar debugging facility.

Using the Debugger

PHP's internal debugger is useful for tracking down evasive bugs. The debugger works by connecting to a TCP port for every time PHP starts up. All error messages from that request will be sent to this TCP connection. This information is intended for "debugging server" that can run inside an IDE or programmable editor (such as Emacs).

How to set up the debugger:

1. Set up a TCP port for the debugger in the configuration file (debugger.port) and enable it (debugger.enabled).
2. Set up a TCP listener on that port somewhere (for example **socket -l -s 1400** on UNIX).
3. In your code, run "debugger_on(*host*)", where *host* is the IP number or name of the host running the TCP listener.

Now, all warnings, notices etc. will show up on that listener socket, *even if you them turned off with error_reporting()*.

Debugger Protocol

The debugger protocol is line-based. Each line has a *type*, and several lines compose a *message*. Each message starts with a line of the type `start` and terminates with a line of the type `end`. PHP may send lines for different messages simultaneously.

A line has this format:

date time
host(*pid*)
type:
message-data

date

Date in ISO 8601 format (*yyyy-mm-dd*)

time

Time including microseconds: *hh*:*mm*:*uuuuuu*

host

DNS name or IP address of the host where the script error was generated.

pid

PID (process id) on *host* of the process with the PHP script that generated this error.

type

Type of line. Tells the receiving program about what it should treat the following data as:

Tabulka D-1. Debugger Line Types

Name	Meaning
`start`	Tells the receiving program that a debugger message starts here. The contents of *data will be the type of error message, listed below.*
`message`	The PHP error message.

Name	Meaning
`location`	File name and line number where the error occured. The first `location` line will always contain the top-level location. *`data will contain file:line. There will always be a location line after message and after every function.`*
`frames`	Number of frames in the following stack dump. If there are four frames, expect information about four levels of called functions. If no "frames" line is given, the depth should be assumed to be 0 (the error occured at top-level).
`function`	Name of function where the error occured. Will be repeated once for every level in the function call stack.
`end`	Tells the receiving program that a debugger message ends here.

`data`

Line data.

Tabulka D-2. Debugger Error Types

Debugger	PHP Internal
warning	E_WARNING
error	E_ERROR
parse	E_PARSE
notice	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
unknown	(any other)

Příklad D-1. Example Debugger Message

1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (null):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice

Dodatek E. PHP reserved words

Here is the list of PHP reserved words and usual constants.

- and.
- break.
- case.
- class.
- continue.
- default.
- do.
- else.
- elseif.
- **empty()**.
- endfor.
- endif.
- endswitch.
- endwhile.
- E_ALL.
- E_PARSE.
- E_ERROR.
- E_WARNING.
- extends.
- FALSE.
- for.
- foreach.
- function.
- if.
- **include()**.
- **include_once()**.
- global.
- **list()**.
- new.
- not.
- or.
- PHP_OS.
- PHP_VERSION.
- **require()**.
- **require_once()**.
- return.
- static.
- switch.
- this.
- TRUE.
- var.

- xor.
- **virtual()**.
- while.
- __FILE__.
- __LINE__.

Dodatek F. PHP’s resource types

Here is the function's list which create, use or destroy PHP resources. You know when a variable is a resource by using **is_resource()**, and what resource type is this variable by using **get_resource_type()**.

Tabulka F-1. Ressource types

Resource type's name	Created by	Used by	Destroyed by	Definition
aspell	**aspell_new()**	**aspell_check(), aspell_check_raw(), aspell_suggest()**	None	Aspell dictionary
bzip2	**bzopen()**	**bzerrno(), bzerror(), bzerrstr(), bzflush(), bzread(), bzwrite()**	**bzclose()**	Bzip2 file
COM	**com_load()**	**com_invoke(), com_propget(), com_get(), com_propput(), com_set(), com_propput()**	None	COM object reference

Resource type's name	Created by	Used by	Destroyed by	Definition
cpdf	**cpdf_open()**	**cpdf_page_init(), cpdf_finalize_page(), cpdf_finalize(), cpdf_output_buffer(), cpdf_save_to_file(), cpdf_set_current_page(), cpdf_begin_text(), cpdf_end_text(), cpdf_show(), cpdf_show_xy(), cpdf_text(), cpdf_set_font(), cpdf_set_leading(), cpdf_set_text_rendering(), cpdf_set_horiz_scaling(), cpdf_set_text_rise(), cpdf_set_text_matrix(), cpdf_set_text_pos(), cpdf_set_text_pos(), cpdf_set_word_spacing(), cpdf_continue_text(), cpdf_stringwidth(), cpdf_save(), cpdf_translate(), cpdf_restore(), cpdf_scale(), cpdf_rotate(), cpdf_setflat(), cpdf_setlinejoin(), cpdf_setlinecap(), cpdf_setmiterlimit(), cpdf_setlinewidth(), cpdf_setdash(), cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto(), cpdf_lineto(), cpdf_rlineto(), cpdf_circle(), cpdf_arc(), cpdf_rect(), cpdf_closepath(), cpdf_stroke(), cpdf_closepath_fill_stroke(), cpdf_fill_stroke(), cpdf_clip(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray_stroke(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor(), cpdf_add_outline(), cpdf_set_page_animation(), cpdf_import_jpeg(), cpdf_place_inline_image(), cpdf_add_annotation()**	**cpdf_close()**	PDF document with CPDF lib

Resource type's name	Created by	Used by	Destroyed by	Definition
cpdf outline				
curl	**curl_init()**	**curl_init(), curl_exec()**	**curl_close()**	Curl session
dbm	**dbmopen()**	**dbmexists(), dbmfetch(), dbminsert(), dbmreplace(), dbmdelete(), dbmfirstkey(), dbmnextkey()**	**dbmclose()**	Link to DBM database
dba	**dba_popen()**	**dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()**	**dba_close()**	Link to DBA base
dba persistent	**dba_open()**	**dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()**	None	Persistant link to DBA base
dbase	**dbase_open()**	**dbase_pack(), dbase_add_record(), dbase_replace_record(), dbase_delete_record(), dbase_get_record(), dbase_get_record_with_names(), dbase_numfields(), dbase_numrecords()**	**dbase_close()**	Link to Dbase base
domxml document				
domxml node				
domxml attribute				
xpath context				
xpath object				
fdf	**fdf_open()**	**fdf_create(), fdf_save(), fdf_get_value(), fdf_set_value(), fdf_next_field_name(), fdf_set_ap(), fdf_set_status(), fdf_get_status(), fdf_set_file(), fdf_get_file(), fdf_set_flags(), fdf_set_opt(), fdf_set_submit_form_action(), fdf_set_javascript_action()**	**fdf_close()**	FDF File

Resource type's name	Created by	Used by	Destroyed by	Definition
ftp	**ftp_connect()**	**ftp_login(), ftp_pwd(), ftp_cdup(), ftp_chdir(), ftp_mkdir(), ftp_rmdir(), ftp_nlist(), ftp_rawlist(), ftp_systype(), ftp_pasv(), ftp_get(), ftp_fget(), ftp_put(), ftp_fput(), ftp_size(), ftp_mdtm(), ftp_rename(), ftp_delete(), ftp_site()**	**ftp_quit()**	FTP stream
gd	**imagecreate(), imagecreatefromgif(), imagecreate-fromjpeg(), imagecreate-frompng(), imagecreate-fromwbmp(), imagecreatefrom-string()**	**imagearc(), imagechar(), imagecharup(), imagecolorallocate(), imagecolorat(), imagecolorclosest(), imagecolorexact(), imagecolorresolve(), imagegammacor-rect(), imagegammacor-rect(), imagecolorset(), im-agecolorsforindex(), imagecolorstotal(), imagecolortranspar-ent(), imagecopy(), imagecopyresized(), imagedashedline(), imagefill(), imagefilledpolygon(), imagefilledrectan-gle(), imagefilltoborder(), imagegif(), imagepng(), imagejpeg(), imagewbmp(), imageinterlace(), imageline(), imagepolygon(), imagepstext(), imagerectangle(), imagesetpixel(), imagestring(), imagestringup(), imagesx(), imagesy(), imagettftext()**	**imagedestroy()**	GD Image
gd font	**imageloadfont()**	**imagechar(), imagecharup(), imagefontheight()**	None	Font for GD

Resource type's name	Created by	Used by	Destroyed by	Definition
gd PS font	**imagepsloadfont()**	**imagepstext(), imagepsslantfont(), imagepsextendfont(), imagepsencodefont(), imagepsbbox()**	**imagepsfreefont()**	PS font for GD
gd PS encoding				
GMP integer	**gmp_init()**	**gmp_intval(), gmp_strval(), gmp_add(), gmp_sub(), gmp_mul(), gmp_div_q(), gmp_div_r(), gmp_div_qr(), gmp_div(), gmp_mod(), gmp_divexact(), gmp_cmp(), gmp_neg(), gmp_abs(), gmp_sign(), gmp_fact(), gmp_sqrt(), gmp_sqrtrm(), gmp_perfect_square(), gmp_pow(), gmp_powm(), gmp_prob_prime(), gmp_gcd(), gmp_gcdext(), gmp_invert(), gmp_legendre(), gmp_jacobi(), gmp_random(), gmp_and(), gmp_or(), gmp_xor(), gmp_setbit(), gmp_clrbit(), gmp_scan0(), gmp_scan1(), gmp_popcount(), gmp_hamdist()**	None	GMP Number

Resource type's name	Created by	Used by	Destroyed by	Definition
hyperwave link	**hw_connect()**	**hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()**	**hw_close(), hw_free_document()**	Link to Hyperwave server

Resource type's name	Created by	Used by	Destroyed by	Definition
hyperwave link persistent	**hw_pconnect()**	**hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()**	None	Persistant link to Hyperwave server

Resource type's name	Created by	Used by	Destroyed by	Definition
hyperwave document	**hw_cp(), hw_docbyanchor(), hw_getremote(), hw_getremotechildren()**	**hw_children(), hw_childrenobj(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getsrcbydestobj(), hw_getandlock(), hw_gettext(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_inscoll(), hw_pipedocument(), hw_unlock()**	**hw_deleteobject()**	Hyperwave object
icap	**icap_open()**	**icap_fetch_event(), icap_list_events(), icap_store_event(), icap_snooze(), icap_list_alarms(), icap_delete_event()**	**icap_close()**	Link to icap server

Resource type's name	Created by	Used by	Destroyed by	Definition
imap	**imap_open()**	**imap_append(), imap_body(), imap_check(), imap_createmailbox(), imap_delete(), imap_deletemailbox(), imap_expunge(), imap_fetchbody(), imap_fetchstructure(), imap_headerinfo(), imap_header(), imap_headers(), imap_listmailbox(), imap_getmailboxes(), imap_get_quota(), imap_status(), imap_listsubscribed(), imap_set_quota(), imap_set_quota(), imap_getsubscribed(), imap_mail_copy(), imap_mail_move(), imap_num_msg(), imap_num_recent(), imap_ping(), imap_renamemailbox(), imap_reopen(), imap_subscribe(), imap_undelete(), imap_unsubscribe(), imap_scanmailbox(), imap_mailboxmsginfo(), imap_fetchheader(), imap_uid(), imap_msgno(), imap_search(), imap_fetch_overview()**	**imap_close()**	Link to IMAP, POP3 serveur
imap persistent				
imap chain persistent				
ingres	**ingres_connect()**	**ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(), ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()**	**ingres_close()**	Persistant link to ingresII base

Resource type's name	Created by	Used by	Destroyed by	Definition
ingres persistent	**ingres_pconnect()**	**ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(), ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()**	None	Link to ingresII base
interbase result	**ibase_query()**	**ibase_fetch_row(), ibase_fetch_object(), ibase_field_info(), ibase_num_fields()**	**ibase_free_result()**	Interbase Result
interbase query	**ibase_prepare()**	**ibase_execute()**	**ibase_free_query()**	Interbase query
interbase blob				
interbase link	**ibase_connect()**	**ibase_query(), ibase_prepare(), ibase_trans()**	**ibase_close()**	Link to Interbase database
interbase link persistent	**ibase_pconnect()**	**ibase_query(), ibase_prepare(), ibase_trans()**	None	Persistant link to Interbase database
interbase transaction	**ibase_trans()**	**ibase_commit()**	**ibase_rollback()**	Interbase transaction
java				
ldap result	**ldap_read()**	**ldap_add(), ldap_compare(), ldap_bind(), ldap_count_entries(), ldap_delete(), ldap_errno(), ldap_error(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_get_option(), ldap_list(), ldap_modify(), ldap_mod_add(), ldap_mod_replace(), ldap_next_attribute(), ldap_next_entry(), ldap_mod_del(), ldap_set_option(), ldap_unbind()**	**ldap_free_result()**	ldap search result

Resource type's name	Created by	Used by	Destroyed by	Definition
ldap link	**ldap_connect(), ldap_search()**	**ldap_count_entries(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_next_attribute(), ldap_next_entry()**	**ldap_close()**	ldap connexion
mcal	**mcal_open(), mcal_popen()**	**mcal_create_calendar(), mcal_rename_calendar(), mcal_rename_calendar(), mcal_delete_calendar(), mcal_fetch_event(), mcal_list_events(), mcal_append_event(), mcal_store_event(), mcal_delete_event(), mcal_list_alarms(), mcal_event_init(), mcal_event_set_category(), mcal_event_set_title(), mcal_event_set_description(), mcal_event_set_start(), mcal_event_set_end(), mcal_event_set_alarm(), mcal_event_set_class(), mcal_next_recurrence(), mcal_event_set_recur_none(), mcal_event_set_recur_daily(), mcal_event_set_recur_weekly(), mcal_event_set_recur_monthly_mday(), mcal_event_set_recur_monthly_wday(), mcal_event_set_recur_yearly(), mcal_fetch_current_stream_event(), mcal_event_add_attribute(), mcal_expunge()**	**mcal_close()**	Link to calendar server

Resource type's name	Created by	Used by	Destroyed by	Definition
msql query	**msql_query()**	**msql(), msql_affected_rows(), msql_data_seek(), msql_dbname(), msql_fetch_array(), msql_fetch_field(), msql_fetch_object(), msql_fetch_row(), msql_fieldname(), msql_field_seek(), msql_fieldtable(), msql_fieldtype(), msql_fieldflags(), msql_fieldlen(), msql_num_fields(), msql_num_rows(), msql_numfields(), msql_numrows(), msql_result()**	**msql_free_result(), msql_free_result()**	mSQL result
msql link	**msql_connect()**	**msql(), msql_create_db(), msql_createdb(), msql_drop_db(), msql_drop_db(), msql_select_db(), msql_select_db()**	**msql_close()**	Link to mSQL database
msql link persistent	**msql_pconnect()**	**msql(), msql_create_db(), msql_createdb(), msql_drop_db(), msql_drop_db(), msql_select_db(), msql_select_db()**	None	Persistant link to mSQL
mssql result	**mssql_query()**	**mssql_data_seek(), mssql_fetch_array(), mssql_fetch_field(), mssql_fetch_object(), mssql_fetch_row(), mssql_field_length(), mssql_field_name(), mssql_field_seek(), mssql_field_type(), mssql_num_fields(), mssql_num_rows(), mssql_result()**	**mssql_free_result()**	Microsft SQL Server result
mssql link	**mssql_connect()**	**mssql_query(), mssql_select_db()**	**mssql_close()**	Link to Microsft SQL Server database
mssql link persistent	**mssql_pconnect()**	**mssql_query(), mssql_select_db()**	None	Persistant link to Microsft SQL Server

Resource type's name	Created by	Used by	Destroyed by	Definition
mysql result	**mysql_db_query(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query()**	**mysql_data_seek(), mysql_db_name(), mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_field(), mysql_fetch_lengths(), mysql_fetch_object(), mysql_fetch_row(), mysql_fetch_row(), mysql_field_flags(), mysql_field_name(), mysql_field_len(), mysql_field_seek(), mysql_field_table(), mysql_field_type(), mysql_num_fields(), mysql_num_rows(), mysql_result(), mysql_tablename()**	**mysql_free_result()**	MySQL result
mysql link	**mysql_connect()**	**mysql_affected_rows(), mysql_change_user(), mysql_create_db(), mysql_data_seek(), mysql_db_name(), mysql_db_query(), mysql_drop_db(), mysql_errno(), mysql_error(), mysql_insert_id(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename()**	**mysql_close()**	Link to MySQL database
mysql link persistent	**mysql_pconnect()**	**mysql_affected_rows(), mysql_change_user(), mysql_create_db(), mysql_data_seek(), mysql_db_name(), mysql_db_query(), mysql_drop_db(), mysql_errno(), mysql_error(), mysql_insert_id(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename()**	None	Persistant link to MySQL database

Resource type's name	Created by	Used by	Destroyed by	Definition
oci8 statement	**ocinewdescriptor()**	**ocirollback(), ocinewdescriptor(), ocirowcount(), ocidefinebyname(), ocibindbyname(), ociexecute(), ocinumcols(), ociresult(), ocifetch(), ocifetchinto(), ocifetchstatement(), ocicolumnisnull(), ocicolumnname(), ocicolumnsize(), ocicolumntype(), ocistatementtype(), ocierror()**	**ocifreestatement()**	Oracle Cursor
oci8 connection	**ocilogon(), ociplogon(), ocinlogon()**	**ocicommit(), ociserverversion(), ocinewcursor(), ociparse(), ocierror()**	**ocilogoff()**	Link to Oracle database
oci8 descriptor				
oci8 server				
oci8 session				
odbc result	**odbc_prepare()**	**odbc_binmode(), odbc_cursor(), odbc_execute(), odbc_fetch_into(), odbc_fetch_row(), odbc_field_name(), odbc_field_num(), odbc_field_type(), odbc_field_len(), odbc_field_precision(), odbc_field_scale(), odbc_longreadlen(), odbc_num_fields(), odbc_num_rows(), odbc_result(), odbc_result_all(), odbc_setoption()**	**odbc_free_result()**	ODBC result

Resource type's name	Created by	Used by	Destroyed by	Definition
odbc link	**odbc_connect()**	**odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()**	**odbc_close()**	Link to ODBC database
odbc link persistent	**odbc_connect()**	**odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()**	None	Persistant link to ODBC database
velocis link				
velocis result				
OpenSSL key	**openssl_get_privatekey(), openssl_get_publickey()**	**openssl_sign(), openssl_seal(), openssl_open(), openssl_verify()**	**openssl_free_key()**	OpenSSL key
OpenSSL X.509	**openssl_x509_read()**	**openssl_x509_parse(), openssl_x509_checkpurpose()**	**openssl_x509_free()**	Public Key

Resource type's name	Created by	Used by	Destroyed by	Definition
oracle cursor	**ora_open()**	**ora_bind(), ora_columnname(), ora_columnsize(), ora_columntype(), ora_error(), ora_errorcode(), ora_exec(), ora_fetch(), ora_fetch_into(), ora_getcolumn(), ora_numcols(), ora_numrows(), ora_parse()**	**ora_close()**	Oracle cursor
oracle link	**ora_logon()**	**ora_do(), ora_error(), ora_errorcode(), ora_rollback(), ora_commitoff(), ora_commiton(), ora_open(), ora_commit()**	**ora_logoff()**	Link to oracle database
oracle link persistent	**ora_plogon()**	**ora_do(), ora_error(), ora_errorcode(), ora_rollback(), ora_commitoff(), ora_commiton(), ora_open(), ora_commit()**	None	Persistant link to oracle database
pdf image	**pdf_open_image(), pdf_open_image_file(), pdf_open_memory_image()**	**pdf_get_image_height(), pdf_get_image_width(), pdf_open_CCITT(), pdf_place_image()**	**pdf_close_image()**	Image in PDF file
pdf outline				

Resource type's name	Created by	Used by	Destroyed by	Definition
pdf document	**pdf_new()**	**pdf_add_bookmark(), pdf_add_launchlink(), pdf_add_locallink(), pdf_add_note(), pdf_add_pdflink(), pdf_add_weblink(), pdf_arc(), pdf_attach_file(), pdf_begin_page(), pdf_circle(), pdf_clip(), pdf_closepath(), pdf_closepath_fill_stroke(), pdf_closepath_stroke(), pdf_concat(), pdf_continue_text(), pdf_curveto(), pdf_end_page(), pdf_endpath(), pdf_fill(), pdf_fill_stroke(), pdf_findfont(), pdf_get_buffer(), pdf_get_image_height(), pdf_get_image_width(), pdf_get_parameter(), pdf_get_value(), pdf_lineto(), pdf_moveto(), pdf_open_ccitt(), pdf_open_file(), pdf_open_image_file(), pdf_place_image(), pdf_rect(), pdf_restore(), pdf_rotate(), pdf_save(), pdf_scale(), pdf_setdash(), pdf_setflat(), pdf_setfont(), pdf_setgray(), pdf_setgray_fill(), pdf_setgray_stroke(), pdf_setlinecap(), pdf_setlinejoin(), pdf_setlinewidth(), pdf_setmiterlimit(), pdf_setpolydash(), pdf_setrgbcolor(), pdf_setrgbcolor_fill(), pdf_setrgbcolor_stroke(), pdf_set_border_color(), pdf_set_border_dash(), pdf_set_border_style(), pdf_set_char_spacing(), pdf_set_duration(), pdf_set_font(), pdf_set_horiz_scaling(), pdf_set_parameter(), pdf_set_text_pos(),**	**pdf_close(), pdf_delete()**	PDF document

Resource type's name	Created by	Used by	Destroyed by	Definition
pgsql link	**pg_connect()**	**pg_cmdtuples(), pg_dbname(), pg_end_copy(), pg_errormessage(), pg_host(), pg_locreate(), pg_loexport(), pg_loimport(), pg_loopen(), pg_lounlink(), pg_options(), pg_port(), pg_put_line(), pg_set_client_encoding(), pg_client_encoding(), pg_trace(), pg_untrace(), pg_tty()**	**pg_close()**	Link to PostGreSQL database
pgsql link persistent	**pg_pconnect()**	**pg_cmdtuples(), pg_dbname(), pg_end_copy(), pg_errormessage(), pg_host(), pg_locreate(), pg_loexport(), pg_loimport(), pg_loopen(), pg_lounlink(), pg_options(), pg_port(), pg_put_line(), pg_set_client_encoding(), pg_client_encoding(), pg_trace(), pg_untrace(), pg_tty()**	None	Persistant link to PostGreSQL database
pgsql result	**pg_exec()**	**pg_fetch_array(), pg_fetch_object(), pg_fieldisnull(), pg_fetch_row(), pg_fieldname(), pg_fieldnum(), pg_fieldprtlen(), pg_fieldsize(), pg_fieldtype(), pg_getlastoid(), pg_numfields(), pg_result(), pg_numrows()**	**pg_freeresult()**	PostGreSQL result
pgsql large object	**pg_getlastoid(), pg_loimport(), pg_loimport()**	**pg_loopen(), pg_getlastoid(), pg_locreate(), pg_loexport(), pg_loread(), pg_loreadall(), pg_lounlink(), pg_lowrite()**	**pg_loclose()**	PostGreSQL Large Object

Resource type's name	Created by	Used by	Destroyed by	Definition
pgsql string				
printer				
printer pen				
printer font				
printer brush				
pspell	**pspell_new(), pspell_new_config(), pspell_new_personal()**	**pspell_add_to_personal(), pspell_add_to_session(), pspell_check(), pspell_clear_session(), pspell_config_ignore(), pspell_config_mode(), pspell_config_personal(), pspell_config_repl(), pspell_config_runtogether(), pspell_config_save_repl(), pspell_save_wordlist(), pspell_store_replacement(), pspell_suggest()**	None	pspell dictionary
pspell config	**pspell_config_create()**	**pspell_new_config()**	None	pspell configuration
Sablotron XSLT	**xslt_create()**	**xslt_closelog(), xslt_openlog(), xslt_run(), xslt_set_sax_handler()**	**xslt_free()**	XSLT parser
shmop	**shm_open()**	**shm_read(), shm_write(), shm_size(), shm_delete()**	**shm_close()**	
sockets file descriptor set	**socket()**	**accept_connect(), bind(), connect(), listen(), read(), write()**	**close()**	Socket
sockets i/o vector				
dir	**dir()**	**readdir(), rewinddir()**	**closedir()**	Dir handle
file	**fopen()**	**feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), flock(), fpassthru(), fputs(), fwrite(), fread(), fseek(), ftell(), fstat(), ftruncate(), set_file_buffer(), rewind()**	**fclose()**	File handle
pipe	**popen()**	**feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()**	**pclose()**	Process handle

Resource type's name	Created by	Used by	Destroyed by	Definition
socket	**fsockopen()**	**fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()**	**fclose()**	Socket handle
sybase-db link	**sybase_connect()**	**sybase_query(), sybase_select_db()**	**sybase_close()**	Link to Sybase Database using DB librairy
sybase-db link persistent	**sybase_pconnect()**	**sybase_query(), sybase_select_db()**	None	Persistant link to Sybase database using DB librairy
sybase-db result	**sybase_query()**	**sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()**	**sybase_free_result()**	Sybase result using DB librairy
sybase-ct link	**sybase_connect()**	**sybase_affected_rows(), sybase_query(), sybase_select_db()**	**sybase_close()**	Link to Sybase Database using CT librairy
sybase-ct link persistent	**sybase_pconnect()**	**sybase_affected_rows(), sybase_query(), sybase_select_db()**	None	Persistant link to Sybase database using CT librairy
sybase-ct result	**sybase_query()**	**sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()**	**sybase_free_result()**	Sybase result using CT librairy
sysvsem	**sem_get()**	**sem_acquire()**	**sem_release()**	System V Semaphore
sysvshm	**shm_attach()**	**shm_remove(), shm_put_var(), shm_get_var(), shm_remove_var()**	**shm_detach()**	System V Shared Memory
wddx	**wddx_packet_start()**	**wddx_add_vars()**	**wddx_packet_end()**	WDDX packet

Resource type's name	Created by	Used by	Destroyed by	Definition
xml	**xml_parser_create()**	**xml_set_object(), xml_set_element_handler(), xml_set_character_data_handler(), xml_set_processing_instruction_handler(), xml_set_default_handler(), xml_set_unparsed_entity_decl_handler(), xml_set_notation_decl_handler(), xml_set_external_entity_ref_handler(), xml_parse(), xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number(), xml_get_current_byte_index(), xml_parse_into_struct(), xml_parser_set_option(), xml_parser_get_option()**	**xml_parser_free()**	XML parser
zlib	**gzopen()**	**gzeof(), gzgetc(), gzgets(), gzgetss(), gzpassthru(), gzputs(), gzread(), gzrewind(), gzseek(), gztell(), gzwrite()**	**gzclose()**	gz-compressed File