



Cultured Perl

Dale Strickland-Clark extols the virtues of Perl, having decided on it as his batch language of choice. He reopens console windows, and makes some selections from his bookshelf.

Following my first couple of Hands On NT columns, when I covered the mixed delights of the console window and DOSKEY macros, I received a number of emails from people asking how to get the macros to load automatically when the console window is created.

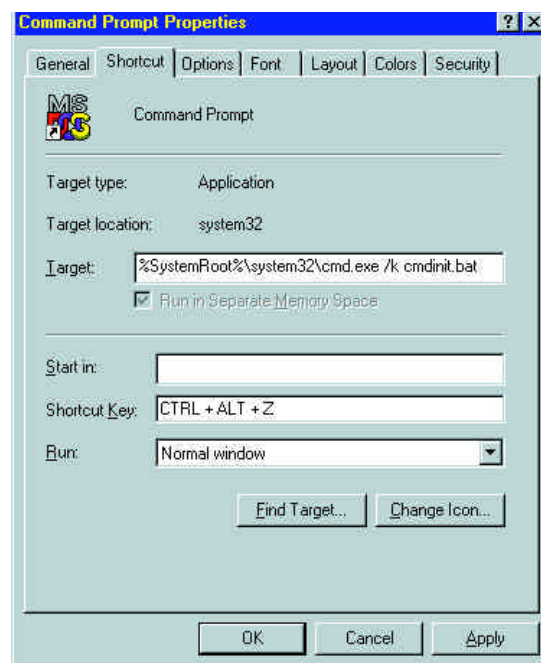
I mentioned at the time that I keep two console sessions running: a small one in the corner of the desktop, and a larger one which spends its time minimised until needed. The larger of the two is started during logon from a shortcut in the StartUp folder. It has the target field set to:

```
%SystemRoot%\system32\cmd.exe /k cmdinit.bat startup
```

and runs minimised. This console window runs the cmdinit.bat (Fig 1) procedure which opens the second and then waits at the command prompt for something else to do.

It's useful to note that we're starting console windows in two slightly different ways here: one from a shortcut and the other from a Start command. If you adjust the properties of a window started from the shortcut (select the window then press ALT-space, P) the changes can be saved back into the shortcut for subsequent uses. Create additional shortcuts when you need windows with different properties.

On the other hand, console windows started with the Start command have no shortcut, so their properties are stored in the Registry and indexed by the window's initial title. Therefore a console started with: `start "Console" cmd.exe` can have a different layout to a window started with:



The Target field contains the command and options necessary to start a console window. Use the shortcut key to make it instantly accessible

applications, just press the shortcut-key combination and a console window should spring into view.

Filename completion

While we're revisiting the console window, here's a handy tip that I don't believe Microsoft has documented anywhere so far. (This isn't available on releases prior to NT 4.)

It's all very well having long, descriptive folder and file names but it means you spend half your day typing path names into console commands. Well, no longer — except, don't attempt this if you're uncomfortable about editing the Registry.

Fire-up the Registry editor (type regedt32 into your nearest console window) and switch to the HKEY_CURRENT_USER window. Locate the Software key, and within that, the Microsoft key. If there isn't already a sub-key called "Command Processor", create one (Edit/Add Key —

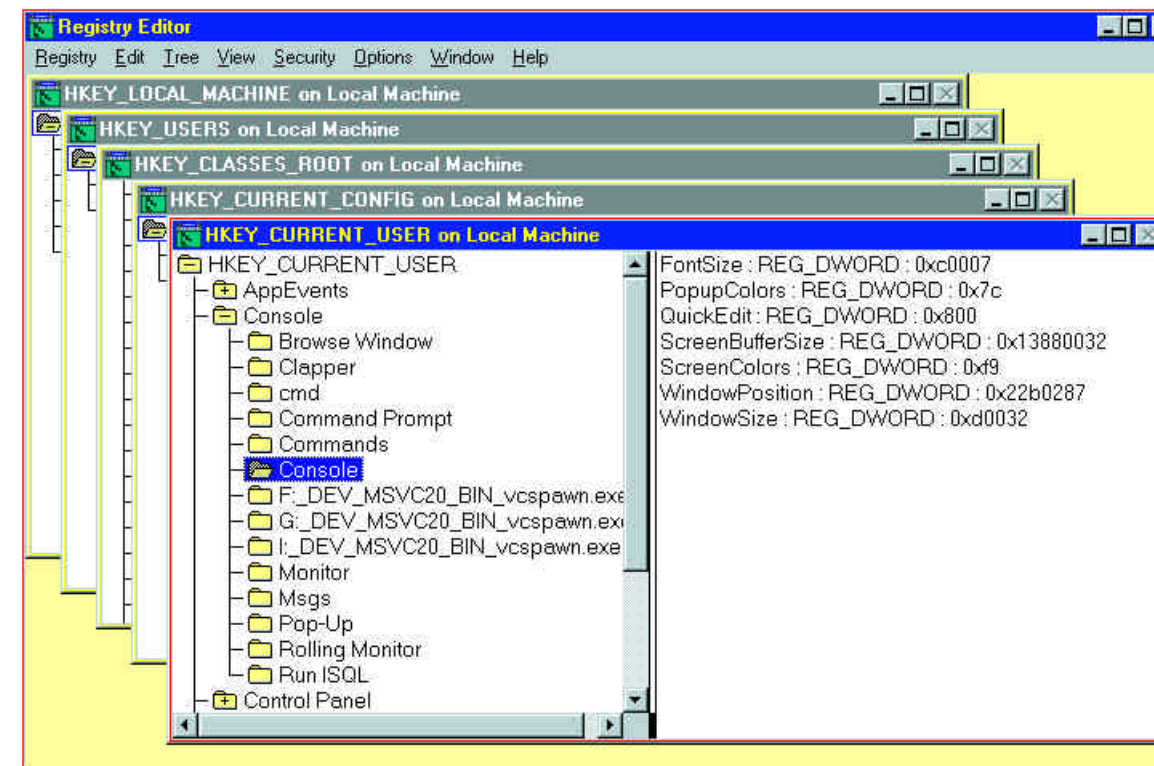
```
start "Demon Status" cmd.exe /k finger status
@gate.demon.co.uk && pause
```

(which Demon Internet users might find useful on a DOSKEY macro).

For those moments when you need a console window quickly, drag a copy of the shortcut from the Startup folder onto your desktop. Edit the parameters in the target field if necessary, set the run style to normal and assign a shortcut-key. Now, from most Windows

Fig 1 Cmdinit.bat

```
@echo off
cd \
doskey /macrofile=c:\batch\macros.txt
prompt $T$H$H$H$H$H$H $P$+$G
if .%1 == .startup start "Console" cmd /k cmdinit.bat
This simple batch file sets the current directory, loads the DOSKEY macros, sets a prompt and finally, if it's been passed the startup parameter, starts the mini console window.
```



Console windows created with a START command have no shortcut so their properties are stored here in the registry

leave the key class blank). Then, within that key, create a value called "CompletionChar" of type REG_DWORD and assign it a value of 9. In other words, set

```
HKEY_CURRENT_USER\Software\Microsoft\Command Processor\CompletionChar = REG_DWORD 0x9
```

Once that's saved, start a new console window. The tab key will now assist by completing partially-entered filenames. If the filename offered isn't the one you want, just press tab again. Press shift-tab to go back through the list of offered names. A partial filename is only recognised if it is at the start of the command or preceded by a space.

If you prefer CTRL-key combinations instead of the tab key, replace the 9 in the registry with 1 for CTRL-A, or 2 for CTRL-B and so on (tab is the same as CTRL-I).

Alternative batch languages

One of the other improvements tucked away in NT 4 is the ability to use alternative batch languages, transparently. The two most popular languages available are probably Rexx (originally from IBM's VM mainframe operating system, subsequently transferred to OS/2) and Perl (a popular Unix shell language much loved by web-site developers). The NT port of Rexx was commissioned by Microsoft to assist users converting from OS/2, and they funded the Perl port to help attract Unix users and capture the web server market.

Fig 2 SearchPath.perl

Finds a file in the path and shows the directory in which found.

```
$target = shift;
print "Looking for $target\n";
for (split /\:/, $ENV{'PATH'}) {
    print "$_";
    print "\\$target <=====" if (-e "$_\\$target");
    print "\n";
}
```

SearchPath.perl — If you've ever wanted to know from which directory a program is being loaded, this little program searches your path and points to the program's home.

Both languages have their strengths: Rexx has a clean, logical, syntax and good string manipulation, while Perl has extensive string manipulation wrapped in a rich, powerful, but less readily-mastered (some might say bizarre) language.

After years as a dedicated Rexx user, I switched to Perl and it's now my batch language of choice, so I'll show the steps required to set it up:

1. Install the Perl system. The latest can be downloaded from ftp.perl.hip.com (see www.perl.hip.com for more information) and I've included a copy on the cover CD. Simply unzip it into the directory where it is to live and run the install.bat procedure.
2. Choose the extension you're going to use for Perl files (I use .perl but .pl is also popular). Create a new environment variable called PATHEXT and assign to it the following string:

```
.com; .exe; .bat; .cmd; .perl
```

Use Control Panel -> System -> Environment to do this permanently.

Enter it into the system or user variables depending on your preference. The order of the extensions listed determines the search order. I've just added .perl to the end of the default value but you can juggle it to suit yourself.

3. Register a file type using the FTYPE command:

```
FTYPE perl file=perl.exe %1 %*
```

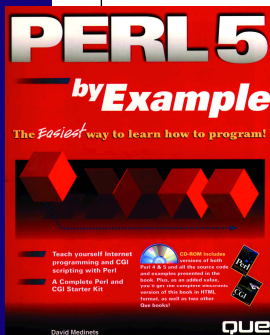
4. Associate the file type with the extension by running the ASSOC command:

```
ASSOC .perl=perl file
```

Note that the FTYPE and ASSOC commands update the Registry and so only need to be run once.

You are now ready to go. Fig 2 is a test program to check your installation and whet your appetite. It scans the directories listed in the search path and shows which contain the file specified as the first parameter.

On the bookshelf



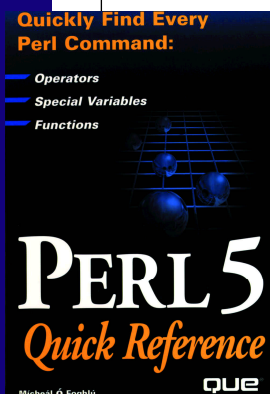
■ Perl 5 by Example
Author David Medinets
Price £37.49
(incl. CD-ROM)
Publisher Que

We've been crying out for a good introductory text to Perl 5 and it has arrived. David Medinets gives the novice and experienced programmer a good understanding of Perl

with only an occasional lapse into that irritating, patronising, tone that spoils many similar programming books. He is clear and logical.

This 658-page book is best treated as a course text, being arranged into increasingly complex subjects, each with its own summary and exercises. The final quarter is devoted to web programming and CGI scripts. The index is good enough to allow you to quickly locate a particular topic.

The CD-ROM is a bit of a mystery. Given that Perl is distributed free under the Gnu artistic licence, you might expect to find various versions for different systems on the CD. But no; there are copies of the code samples from the book, plus texts from a couple of other books and a copy of Microsoft's Internet Explorer. There's a copy of MIRC (Internet chat client), too, whose connection with the book's subject-matter is tenuous to say the least.



■ **Perl 5 Quick Reference**
 Author Micheál Ó Foghlú
 Price £18.49
 Publisher Que

If this book does nothing else, it at least helps end the debate about what works on NT's Perl and what doesn't. With each function listed, there is a set of compliance icons indicating whether the feature works on versions 4, 5 or NT; the

implication being that everything works on Unix.

Overall, it's a bit of a mess. There are a number of typographical errors, the section on pattern matching omits any mention of alternation (the use of the bar, "|" symbol to match either of two patterns), although the information can be found at the back of the book.

But if you have this 345-page book, you will use it. It's a handy reference for your bookshelf. It might have been a handy pocket reference had they not been quite so extravagant with space, which is wasted in abundance. But then it wouldn't have looked like eighteen quid's worth, would it?

Fig 3 ListUser.perl

```
# List all users in the domain along with one selected attribute.
# Subroutine to return an array of all the usernames in the domain
```

```
sub GetUserNames
{
    my $userline = 0;
    map /(\\S.{1,19}?)\\s\\s{1,19}/g, grep {
        if (/^-----/)
        {
            $userline = 1;
            0
        }
        else
        {
            $userline = 0 if /^The command/;
            $userline
        }
    } qx(net user /domain);
}
```

```
# Retrieve parameter or supply default
$property = shift || "Full name":
```

```
# Define output format for report header
```

[illegible]

```
# Define output format for report body
```

[illegible]

```
# Iterate through all the user names extracting the selected attribute.
```

```
for (GetUserNames)
{
    ($line) = grep (/ $property/i, qx(net user "$_" /domain));
    ($propname, $value) = $line =~ /^(\S. +?)\s{2,}(\S*.*)$/;
    write;
}
}
```

ListUser.perl. Lists all the users in your domain with a single specified attribute. The program is rather simple-minded: if you mis-type the attribute name it won't notice; neither will it produce interesting output. The program demonstrates useful techniques, like capturing the output from commands, filtering and extracting information from them and formatting a simple report.

Fig 3, ListUser.perl, is a bit more complex, demonstrating the ability of Perl to achieve a lot with few statements. It's a post-processor for NT's NET USER command and displays the value of a single attribute for each user in a domain. The program takes a single parameter which should be enclosed in double quotes if it contains spaces. The parameter is the name, or part of the name, of the attribute to display. For a list of all the accepted attributes, enter:

```
NET USER <your id> /DOMAIN
```

into a console window. If you're not logged on to a domain, remove the option from the command and both uses of the NET

command in the program.

For example, to list the last logon time for each user, the command is

listuser "last logon"

The GetUserNames subroutine is self-contained and can be plucked out for use in another program or transferred to a function library. (Both the Perl programs listed here are included on our cover-mounted CD.)

- PCW Contacts

Dale Strickland-Clark is a journalist and consultant on Windows/NT and the internet. He can be reached at nt@pcw.vnu.co.uk.
Computer Manuals 0121 706 6000