WinHex 9.72

General Information

Working with the Hex Editor

Menu Reference

Extras

WinHex 9.72

Author: Stefan Fleischmann
E-Mail: mail@sf-soft.com

First released in 1995, last updated January 2001.

The following operating systems are supported:
- Windows 95/98/Me
- Windows NT 4.0
- Windows 2000

Homepage: http://www.winhex.com and http://www.sf-soft.com
Forum: http://www.winhex.net

Please find the latest version of this program there.

French translation by Jérôme Broutin (user interface) and Henri Pouzoullic (program help). Updated by Bernard Leprêtre. Entire Spanish translation by José María Tagarro Martí. Italian translation by Luca Cantarini (user interface). Brazilian Portuguese translation by Heyder Lino Ferreira (user interface). Simplified Chinese (GB) language pack by Lv Darong (available separately).

Cryptography consulting: Alexandre Pukall

ZDNet Software Library Rating: 5 stars (of 5)!

U.S. National Institutes (e.g. the Oak Ridge National Laboratory in Tennessee), the Technical University of Vienna, the Technical University of Munich (Institute of Computer Science), Toshiba Europe, Novell Inc., Ontrack Data International Inc., Siemens Business Services, Mannesmann VDO AG, Password Crackers Inc., DePfa Deutsche Pfandbriefbank AG, Analytik Jena AG, and many other companies and scientific institutes are already registered users. Order the full version now!

Using a Hex Editor

A hex editor is capable of completely displaying the contents of each file type. Unlike a text editor, a hex editor even displays control codes (e. g. linefeed and carriage-return characters) and executable code, using a two-digit number based on the hexadecimal system.

Consider one byte to be a sequence of 8 bits. Each bit is either 0 or 1, it assumes one of two possible states. Therefore one byte can have one of 2•2•2•2•2•2•2•2 = 2^8 = 256 different values. Since 256 is the square of 16, a byte value can be defined by a two-digit number based on the hexadecimal system, where each digit represents a tetrade or nibble of a byte, i. e. 4 bits. The sixteen digits used in the hexadecimal system are 0-9, A-F.

You can change the value of a byte by changing these digits in the hexadecimal mode. It is also possible to enter the character that is assigned to a certain byte value by a character set (cf. Entering Characters). All kinds of characters are allowed (e.g. letters and punctuation marks). Example: A byte whose decimal value is 65 is displayed as 41 in hexadecimal notation (**4**•16+**1**=**65**) and as the letter A in text mode. The ASCII character set defines the capital letter A to have the decimal value of 65.

When editing files of a certain type (for instance executable files), it is essential not to change the file *size*. Moving the addresses of executable code and included data results in severely damaging such files. Please note that changing the contents of a file generally may be the reason for the corresponding application to behave anomalously. It is quite safe to edit text passages in a file. At any rate, it is recommendable to create backup files before editing.

The command "Combined Search" was especially designed for editing files created by computer games to save the game state. If you know the value of a variable in two of such files, you can find out the offset, i. e. the position, at which this data is saved. Example: If two files hold the information that you have 5 resp. 7 points/lives/..., search simultaneously for the hex value 05 in the first and 07 in the second file.

Ordering Information

There are licenses for private use (for non-commercial purposes only, in a non-business, non-institutional, and non-government environment) and for use in a company, in an organization or in public administration. If you are going to use WinHex on multiple machines, you will also need additional licenses.

Pricing

Base license:	EUR 31.90 / US$ 34 (private use) EUR 56.90 / US$ 60 (professional use)
Additional licenses:	EUR 16.90 / US$ 18 each (private use) EUR 29.90 / US$ 32 each (professional use)

• Please visit the WinHex Homepage http://www.winhex.com to learn about how to **order online** and how to pay by **credit card**. *or*

• Recommended within Germany: Transfer the money directly into my bank account (see below) and notify me of this. When transferring from outside of Germany, add EUR 5 / US$ 5. *or*

• Within Europe you may send me a Eurocheque (with the "ec" symbol). *or*

• Send me a (normal) check or a money order. Because of bank charges add EUR 6 / US$ 7. *or*

• Send cash money (on your own risk).

In any of the latter four cases please order via e-mail or conventional mail and specify "WinHex 9.72" and your address.

You need an invoice or a receipt for your company/organization/...? Simply tell me.

After having received the registration fee, you will be send codes that turn WinHex into a full version. This full version will save files larger than 250 KB, write disk sectors, edit virtual memory and show no evaluation version reminders. Later versions that are released within 12 months, counted from the release of this version, are included (possibly more).

Stefan Fleischmann
Carl-Diem-Str. 32
D-32257 Bünde
Germany

Homepage: http://www.winhex.com and http://www.sf-soft.com
E-Mail: mail@sf-soft.com

Please visit my homepage if you are interested in later versions of this program.

Account owner: Stefan Fleischmann
Account No.: 1208127686
Bank: Sparkasse Herford
Code No.: 49450120
Swift code: WLAH DE 44
Bank address: Auf der Freiheit 20, D-32052 Herford (Germany)

Thank you very much!

Integer Data Types

Format/Type	Range	Example
signed 8 bit	-128...127	FF = -1
unsigned 8 bit	0...255	FF = 255
signed 16 bit	-32,768...32,767	00 80 = -32,768
unsigned 16 bit	0...65,535	00 80 = 32,768
signed 32 bit	-2,147,483,648...2,147,483,647	00 00 00 80 = -2,147,483,648
unsigned 32 bit	0...4,294,967,295	00 00 00 80 = 2,147,483,648
signed 64 Bit	-2^63...2^63-1	00 00 00 00 00 00 00 80 = -2^63

Unless stated otherwise, multi-byte numbers are stored in little-endian format, meaning that the first byte of a number is the least significant and the last byte is the most significant. This is the common format for computers running Microsoft Windows. Following the little-endian paradigm, the hexadecimal values 10 27 can be interpreted as the hexadecimal number 2710 (decimal: 10,000).

The Data Interpreter is capable of interpreting data as all of the aforementioned integer types.

Floating-Point Data Types

Type	Range	Precision	Bytes
Float (Single)	± 1.5e-45..3.4e+38	7-8	4
Real	± 2.9e-39..1.7e+38	11-12	6
Double (Double)	± 5.0e-324..1.7e+308	15-16	8
Long Double (Extended)	± 3.4e-4932..1.1e+4932	19-20	10

The type names originate from the C programming language. The corresponding Pascal names are specified in brackets. The Real type exists only in Pascal. The Data Interpreter is capable of translating hex values in an editor window into floating-point numbers of all four types and vice-versa.

In the computer, a floating-point number F is represented by a mantissa M and an exponent E, where M × 2^E = F. Both M and E are signed integer values themselves. The four data types differ in their value ranges (i.e. the number of bits reserved for the exponent) and in their precision (i.e. the number of bits reserved for the mantissa).

On Intel-based systems, calculations upon floating-point numbers are carried out by a math coprocessor while the main processor waits. The Intel 80x87 uses 80-bit precision for calculations, whereas RISC processors often use 64-bit precision.

ANSI ASCII/IBM ASCII

ANSI ASCII is the character set used in Windows applications. It is standardized by the American National Standards Institute. MS-DOS uses the IBM ASCII character set (also called OEM character set). These character sets differ in the second half, containing characters with a ASCII values greater than 127.

It is reasonable to switch the menu option "Use ANSI ASCII" off when viewing or editing files belonging to a DOS program.

Use the "Convert" command of the Edit Menu to convert text files from one character set into the other.

The first 32 ASCII values do not define printable characters, but control codes:

Hex	Control Code
0	Null
1	Start of Header
2	Start of Text
3	End of Text
4	End of Transmission
5	Enquiry
6	Acknowledge
7	Bell
8	Backspace
9	Horizontal Tab
A	Line Feed
B	Vertical Tab
C	Form Feed
D	Carriage Return
E	Shift Out
F	Shift In
10	Data Link Escape
11	Device Control 1 (XON)
12	Device Control 2
13	Device Control 3 (XOFF)
14	Device Control 4
15	Negative Acknowledge
16	Synchronous Idle
17	End of Transmission Block
18	Cancel
19	End of Medium
1A	Substitute
1B	Escape
1C	File Separator
1D	Group Separator
1E	Record Separator
1F	Unit Separator

Checksums

A checksum is a characteristic number used for verification of data authenticity. Two files with equal checksums are highly likely to be equal themselves (byte by byte). Calculating and comparing the checksums of a file *before* and *after* a possibly inaccurate transmission may reveal transmission errors. An unaffected checksum indicates that the files are (in all likelihood) still identical. However, a file can be manipulated on purpose in such a way that its checksum remains unaffected. Digests are used instead of checksums in such a case, where malicious (i.e. not mere random) modifications to the original data are to be detected.

In WinHex, checksums are calculated when opening (optional, cf. General Options) or analyzing (cf. Tools Menu) a file. After modifying files, checksums can be re-calculated by pressing Alt+F2.

The **standard checksum** is simply the sum of all bytes in a file, calculated on a 32-bit accumulator. The **CRC32** (**c**yclic **r**edundancy **c**ode) is based on a more sophisticated algorithm, which is even safer.

Example: If a transmission alters two bytes of a file in such a way that the modifications are countervailing (for instance byte one +1, byte two -1), the standard checksum remains unaffected, whereas the CRC32 changes.

Technical Hints

Amount of memory used:	0.5 MB
~ per routine:	0.5 KB
Maximum number of windows:	1000 (Windows NT/2000), 500 (Windows 9x)
Maximum file & disk size:	~ 1024 GB
Max. no. of program instances:	99
Max. no. of routines:	100
Max. reversible keyboard inputs:	65535
Encryption depth:	128 bit
Digest length in backups:	256 bit
Character sets supported:	ANSI ASCII, IBM ASCII, EBCDIC (limited)
Offset presentation:	hexadecimal/decimal

• In most cases, the progress display shows the completed percentage of an operation. However, during search and replace operations it indicates the relative position in the current file or disk.

• The user interface looks best if *no* extra large font is used in your Windows system.

• WinHex expects your computer to be running in little-endian mode.

• Search and replace operations generally run fastest with case sensitivity switched on and without wildcards enabled.

• Here are some pieces of information concerning the Master Boot Record of a hard disk, that is editable using the Disk Editor.

• When searching with the option "count occurrences" activated or when replacing without prompting, for a search algorithm there are generally two ways to behave when an occurrence has been found, which in some cases may have different results. This is explained by the following example:

The letters "ana" are searched in the word "banana". The first occurrence has already been found at the second character.
1st alternative: The algorithm continues the search at the third character. So "ana" is found again at the fourth character.
2nd alternative: The three letters found in the word "banana" are skipped. The remaining letters "na" do not contain "ana" any more.

WinHex is programmed in the second manner, since this delivers the more reasonable results when counting or replacing occurrences. (If you continue a search using the F3 key or you choose the replace option "prompt when found", the algorithm follows the first alternative.)

• For further technical information, please consult the WinHex homepage at http://www.winhex.com.

Legal Information

1. GRANT OF LICENSE. This EULA grants you the following limited, non- exclusive rights: [...]
* Distribution Requirements. You may copy and redistribute the Sample Code and/or Redistributable Code (collectively "REDISTRIBUTABLE COMPONENTS") as described above, provided that (a) you do not distribute the REDISTRIBUTABLE COMPONENTS preinstalled on any computer or other digital electronic device; (b) the REDISTRIBUTABLE COMPONENTS only operate in conjunction with Microsoft Windows NT 4.0 and Windows 2000; (c) you may only permit further redistribution of the REDISTRIBUTABLE COMPONENTS pursuant to license terms at least as restrictive as those contained in this EULA and in \LICENSE\REDIST.TXT; (d) you do not use Microsoft's name, logo, or trademarks to market your Tool Product; (e) you include a valid copyright notice on your Tool Product and do not remove any copyright notice contained in the REDISTRIBUTABLE COMPONENTS; and (f) you agree to indemnify, hold harmless, and defend Microsoft from and against any claims or lawsuits, including attorneys' fees, that arise or result from the use or distribution of your Tool Product. Contact Microsoft for the applicable royalties due and other licensing terms for all other uses and/or distribution of the REDISTRIBUTABLE COMPONENTS, including preinstallation of the SOFTWARE PRODUCT or portions thereof.
* Microsoft reserves all rights not expressly granted to you.

2. COPYRIGHT. All title, including but not limited to copyrights, in and to the SOFTWARE PRODUCT and any copies thereof are owned by Microsoft or its suppliers. All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content.

3. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.
* Limitations on Reverse-Engineering, Decompilation, and Disassembly. You may not reverse- engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
* Rental. You may not rent, lease or lend the SOFTWARE PRODUCT.
* Support Services. Microsoft may provide you with support services related to the SOFTWARE PRODUCT ("Support Services"). Use of Support Services is governed by the Microsoft policies and programs described in the user manual, in "online" documentation, and/or in other Microsoft-provided materials. Any supplemental software code provided to you as part of the Support Services shall be considered part of the SOFTWARE PRODUCT and subject to the terms and conditions of this EULA. With respect to technical information you provide to Microsoft as part of the Support Services, Microsoft may use such information for its business purposes, including for product support and development. Microsoft will not utilize such technical information in a form that personally identifies you.
* Software Transfer. You may permanently transfer all of your rights under this EULA, provided you retain no copies, you transfer all of the SOFTWARE PRODUCT (including all component parts, the media and printed materials, any upgrades, this EULA, and, if applicable, the Certificate of Authenticity), and the recipient agrees to the terms of this EULA. If the SOFTWARE PRODUCT is an upgrade, any transfer must include all prior versions of the SOFTWARE PRODUCT.
* Termination. Without prejudice to any other rights, Microsoft may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

4. INTELLECTUAL PROPERTY RIGHTS. All ownership, title and intellectual property rights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text and "applets" incorporated into the SOFTWARE DUCT), and any copies you are permitted to make herein are owned by Microsoft or its suppliers. All ownership, title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This EULA grants you no rights to use such content. For each copy of the SOFTWARE PRODUCT you are authorized to use above, you may also reproduce one additional copy of the SOFTWARE PRODUCT solely for archival or restoration purposes.

5. U.S. GOVERNMENT RESTRICTED RIGHTS. All SOFTWARE PRODUCT provided to the U.S.

Government pursuant to solicitations issued on or after December 1, 1995 is provided with the commercial rights and restrictions described elsewhere herein. All SOFTWARE PRODUCT provided to the U.S. Government pursuant to solicitations issued prior to December 1, 1995 is provided with RESTRICTED RIGHTS as provided for in FAR, 48 CFR 52.227-14 (JUNE 1987) or DFAR, 48 CFR 252.227-7013 (OCT 1988), as applicable.

6. EXPORT RESTRICTIONS. You agree that you will not export or re-export the SOFTWARE PRODUCT, any part thereof, or any process or service that is the direct product of the SOFTWARE PRODUCT (the foregoing collectively referred to as the "Restricted Components"), to any country, person or entity subject to U.S. export restrictions. You specifically agree not to export or re-export any of the Restricted Components (i) to any country to which the U.S. has embargoed or restricted the export of goods or services, which currently include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria, or to any national of any such country, wherever located, who intends to transmit or transport the Restricted Components back to such country; (ii) to any person or entity who you know or have reason to know will utilize the Restricted Components in the design, development or production of nuclear, chemical or biological weapons; or (iii) to any person or entity who has been prohibited from participating in U.S. export transactions by any federal agency of the U.S. government. You warrant and represent that neither the U.S. Commerce Department, Bureau of Export Administration nor any other U.S. federal agency has suspended, revoked or denied your export privileges.

7. APPLICABLE LAW. If you acquired the SOFTWARE PRODUCT in the United States, this EULA is governed by the laws of the State of Washington. If you acquired the SOFTWARE PRODUCT in Canada, unless expressly prohibited by local law, this EULA is governed by the laws in force in the Province of Ontario, Canada; and, in respect of any dispute which may arise hereunder, you consent to the jurisdiction of the federal and provincial courts sitting in Toronto, Ontario. If the SOFTWARE PRODUCT was acquired outside the United States, then local law may apply.

8. QUESTIONS. Should you have any questions concerning this EULA, or if you desire to contact Microsoft for any reason, please contact the Microsoft subsidiary serving your country, or write: Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399.

9. DISCLAIMER OF WARRANTIES. To the maximum extent permitted by applicable law, Microsoft and its suppliers provide the SOFTWARE PRODUCT and any (if any) support services related to the SOFTWARE PRODUCT ("Support Services") AS IS AND WITH ALL FAULTS, and hereby disclaim all warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties or conditions of merchantability, of fitness for a particular purpose, of lack of viruses, of accuracy or completeness of responses, of results, of workmanlike effort, and of lack of negligence, all with regard to the SOFTWARE PRODUCT, and the provision of or failure to provide Support Services. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE PRODUCT. THE ENTIRE RISK AS TO THE QUALITY OF OR ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT AND SUPPORT SERVICES, IF ANY, REMAINS WITH YOU.

10.EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND

EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

11. LIMITATION OF LIABILITY AND REMEDIES. NOTWITHSTANDING ANY DAMAGES THAT YOU MIGHT INCUR FOR ANY REASON WHATSOEVER (INCLUDING, WITHOUT LIMITATION, ALL DAMAGES REFERENCED ABOVE AND ALL DIRECT OR GENERAL DAMAGES), THE ENTIRE LIABILITY OF MICROSOFT AND ANY OF ITS SUPPLIERS UNDER ANY PROVISION OF THIS EULA AND YOUR EXCLUSIVE REMEDY FOR ALL OF THE FOREGOING SHALL BE LIMITED TO THE GREATER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT OR U.S. $5.00. THE FOREGOING LIMITATIONS, EXCLUSIONS AND DISCLAIMERS SHALL APPLY TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, EVEN IF ANY REMEDY FAILS ITS ESSENTIAL PURPOSE.

General Options

1st column:

• When selecting **WinHex as default association**, the registry key HKEY_CLASSES_ROOT\Unknown\shell\Open\Command is modified. This enables you to open a file of an unregistered file type in WinHex by double-clicking it in the Explorer window. This works as well with unregistered and even executable files if you press the Shift key.

• You may have **WinHex** appear in the Windows **context menu**. The shell displays the context menu when the user clicks an object with the right mouse button. WinHex provides menu items for files, folders and disks. If this option is not fully selected, there is no menu item for files.

• The option **Allow multiple program instances** lets you execute WinHex more than once on a single computer at a time. If it is not enabled, WinHex puts the main window of the running instance into the foreground instead of creating a new program instance.

• **Do not update file time** means that WinHex will not change the last write time when a modified file is saved.

• Specify the **number of recently opened files** to be listed at the end of the File Menu.

• The **toolbar** is displayed optionally.

• A **tab control** makes each edit window accessible with a single mouse click only.

• The **details panel** provides in-depth information on any open object (file, disk, RAM).

• If you select **Show file icons**, the icons stored in a file are shown in the information column. This increases memory requirements and delays the opening of files. If a file contains no icons, the icon of the file *type* is shown, except this option is not "fully" selected.

• The **ENTER** key can be used to enter up to four two-digit hex values. A useful example is **0x0D0A**, which is interpreted as an end-of-line marker in the Windows world (Unix: 0x0D).

• Decide whether you want to use the **TAB key** to switch from text to hexadecimal mode and vice versa or to enter the TAB character (0x09). In any case, TAB+SHIFT can be pressed to switch the current mode.

2nd column:

• Specify the **folder** in which to create **temporary files**.

• Specify the **folder** in which to create **backup files**.

• Specify a path to your favorite **text editor**. WinHex offers you to view the report of file comparisons using this program. The standard Windows editor Notepad is not capable of opening files larger than 64 KB.

• **Offsets** can be presented and prompted for in a decimal or **hexadecimal** notation. This setting is valid for the entire program.

• When using the RAM editor it may be reasonable to have WinHex display **virtual addresses** instead of zero-based offsets. This is always done in hexadecimal notation. The dialog window of the Goto Offset command will also prompt for virtual addresses.

• If you choose **Show 0x01-0x1F as dots**, hex values in the range from 01 to 1F will be displayed as a dot in text mode when using the ANSI ASCII character set.

• Decide the number of **bytes per line** in an edit window. Common values are 16 or 32 (depending on the screen resolution).

• Decide how many **bytes** shall be displayed in a **group**. Powers of 2 serve best for most purposes.

3rd column:

• By default, WinHex displays a **double cursor**. There are two kinds of secondary cursors available.

• If line-by-line scrolling is selected, **page** and sector **separators** may be **display**ed. If this option is enabled partially, only sector separators are displayed.

• **Scrolling** can be done either **line by line** or pagewise.

• Specify how many **lines to scroll** when **roll**ing the mouse **wheel** (if available).

• By default, **edit windows** are **opened** in a **maximized** state.

• **Use Windows default colors** should be self-explanatory.

• Select a **color** used as the **background** of the current **block**. You can only change the color if the option "Use Windows default colors" is switched off.

• You may choose a **font** for ANSI ASCII mode. The WinHex font implements the full Windows character set (even characters such as the TM and Euro symbols and diverse quotation marks).

• Decide whether WinHex shall **Play message sounds** when showing a hints or questions.

• **Display Windows-styled progress bar** replaces the WinHex progress bar with the typical progress bar common to most Windows programs.

• You may select one of four different **dialog window styles**.

--

Factory settings of *all* options can be restored using the Initialize command of the Help menu.

Entering Characters

In hex mode only hexadecimal characters are to be entered ('0'...'9', 'A'...'F'). In text mode you can enter all kinds of characters: letters, numbers, punctuation marks and special characters (e. g. '»', ']' and '^'). Please use the Windows program charmap.exe to find out key combinations for such characters (e. g. Alt-1-7-5 for '»'). The "WinHex" font even supports the Euro symbol.

Edit Modes

Default edit mode: Modifications to files or disks opened in default edit mode are stored in temporary files. The latter are either created dynamically when needed or when opening the original file (cf. option "open files quickly). The menu command "Save" of the File Menu updates the original file or disk.

View mode: Files or disks that are opened in view mode cannot be edited, only viewed. In other words, they are opened write-protected.

In-place edit mode: Please use caution when opening files or disks in in-place edit mode. *All* kinds of modifications (keyboard input, filling/removing the block, writing clipboard data, replacements, ...) are written *to the original file* ("in-place") *without prompting*! It is not necessary to save the file manually after having modified it. Instead, the modifications are saved lazily and automatically, at latest when closing the edit window. However, you may use the Save command to ensure the buffer is flushed at a given time.

The in-place edit mode is preferable if the data transfer from the original to the temporary file and vice-versa, which is obligatory in default edit mode for certain operations, consumed too much time or disk space. This may be the case when opening very large files or when modifying huge amounts of data. Since usually no temporary files are needed in in-place edit mode, this edit mode is generally faster than the default edit mode. The in-place edit mode is the only mode available when using the RAM editor.

Hint: Even in in-place edit mode the creation of a temporary file is unavoidable when altering the file *size*.

Status Bar

The status bar displays the following information about a file:

1. Number of current page and total number of pages (disk editor: sectors)
2. Current position (offset)
3. Decimal translation of the hex values at the current position
4. Beginning and end of the current block (if currently defined)
5. Size of current block in bytes (ditto)

Click the status bar cells in order to...
1. Move to another page/sector,
2. Move to another offset,
3. Define the integer type for decimal translation and
4. Define the block.

Right-click the status bar in order to copy pieces of information from the status bar into the clipboard.

Right-clicking the 2nd status bar field permits switching between absolute (default) and relative offset presentation. This is useful when examining data that consists of records of a fixed length. After specifying the record length in bytes, the status bar displays the current record number and the relative offset therein.

Right-clicking the 3rd status bar field also permits copying the four hex values at the current position in reverse order into the clipboard. This is useful for following pointers.

Useful Hints

• Use the mouse buttons to define the block. Double-clicking the right button clears the block.
• You may want to define the block using the keyboard (Shift+arrow keys or Alt+1 and Alt+2).
• Use the TAB key to switch between hexadecimal and text mode.
• Ctrl+Q closes all windows.
• (Ctrl+)Enter displays the Window Manager.
• ESC aborts the current operation if any, otherwise removes the current block selection.
• PAUSE stops or continues the current operation.
• F11 repeats the last Go To Offset command.
• Shift+F7 switches between three character sets.
• (Shift+)Alt+F11 repeats the last Move Block command.
• Alt+F2 recalculates the checksum after a file was modified.

• Enable the first two general options to integrate WinHex into the Windows shell.

• WinHex accepts filenames specified in the command line, and is drag-and-drop capable.

• Use routines to make your work with WinHex more efficient.

• You can specify the ID number of a routine as a parameter in order to execute the routine.

• Switch from hexadecimal to decimal offset presentation by clicking the offset numbers.

• If a screen resolution of 800×600 or more is provided you can increase or decrease the editor window and the information column at the right by clicking and pulling the bottom borders.

• Try clicking the status bar cells (left and right mouse button).

Routines

It may be recommendable to create a routine once rather than making the same changes to a file over and over again. WinHex is capable of maintaining up to 100 routines, stored in the file *routines.dat*. The Routine Manager allows you to create, edit, copy and delete routines.

1. In order to create a routine you must specify which file(s) the routine shall apply to. Either the routine applies to the current file, to all open files or to an existing file on a disk. A routine cannot be applied to a file opened in view mode.

2. You may directly address and change up to 5 byte values. For each specify a two-digit hex value and the offset.

3. A routine can be used to replace text or hex values just as the corresponding menu commands (cf. Replace Options). The scope is the entire file unless start and end offsets are specified.

4. Among other options, you may specify a routine that is to be executed when the current routine has completed. To this routine, the current file is the one to which the previous routine was applied. The routine next to a routine which applies to all opened files must not apply to a "current" file, since at runtime it is unknown which file is meant.

Hints:

• WinHex executes a routine automatically if you specify the ID number of that routine as a parameter (e.g. "winhex 4").

• Such a routine may apply to "all open files", if WinHex is already running with at least one file opened or the filenames are specified as precedent parameters.

• When executing several routines that apply to "all open files", only the last one needs to run with the option "save changes automatically" enabled (if desired to apply to all open files).

• If the last parameter is "auto", WinHex terminates automatically after executing the specified routines.

Example:

Consider two routines being defined (Nos. 1 and 2), that both apply to "all open files". Routine No. 2 is configured to save changes automatically. The command line "winhex c:\file1.dat 1 d:\file2.dat 2 auto" has the following result: At first, the file "c:\file1.dat" is opened. Then the routine with the ID number 1 is applied to this file. After that, "d:\file2.dat" is opened, routine No. 2 is applied to both files, and changes are written to the disk. Finally, WinHex is terminated.

Disk Editor

The Disk Editor, that is part of the Tools menu, enables you to access floppy and hard disks below file-system level. Disks consist of sectors (commonly units of 512 bytes). You may access a disk either logically (i. e. controlled by the operating system) or physically (controlled by the BIOS). On most computer systems you can even access CD-ROM and DVD media.

Questions & Answers

Please note the following limitations:

• Under Windows NT, administrator rights are needed to access hard disks.
• Replace functions are not available.
• WinHex cannot *write* to CD-ROM or DVD.
• The disk editor cannot operate on remote (network) drives.

Edit free space on drive (Windows 95/98)

Under Windows 95/98, it is possible to edit the currently unused space on a logical disk. Thethe aforementioned limitations do not apply in this case. WinHex creates a file which uses the complete free space on the selected drive. You can edit this file in in-place mode. The integrity of data in the used parts of the drive *cannot be affected* hereby.

Application examples *for Windows 95/98*:
• You can use this function to recover unintentionally deleted data which has not yet been overwritten by new files. Search for the data, mark it as the current block and copy it.
• This function is useful to "wipe" unused space on a drive for security reasons. Confidential information is possibly stored in currently unused parts of a drive as a result of normal delete, copy and save actions. Simply define the whole file as the block and fill it with the hex value 00.

Initialize free space on drive (Windows NT)

Under Windows NT, free space on a drive can be *initialized with zero bytes*, e. g. for security reasons. This effectively deletes all data in unused parts of the disk and makes it impossible to recover this data.

Annotation: Of course, data that has been deleted by WinHex using the Delete irreversibly command cannot be found in unused parts of a drive any more.

Save Sectors: To be used analogously to the Save command for files. Part of the File menu. Writes all modifications to the disk. Please note that, depending on your changes, this may severely damage the integrity of the disk data. If the corresponding undo option is enabled, a backup of the concerned sectors is created, before they are overwritten.
The command can only be used in the full version.

Here are some pieces of information concerning the Master Boot Record of a hard disk, that is editable using the disk editor.

File Menu

New: This command is used to create a file. The file is initialized with zero bytes and principally opened in default edit mode. You have to specify the desired file size.

Open: Lets you open one or more files. You may choose an edit mode in case it is not predetermined in the Tools menu.

Save: Saves the currently displayed file with all modifications to the disk. In in-place edit mode, using this command is not necessary. When using the disk editor, this command is named "Save Sectors".

Save As: Saves the currently displayed file under a different name.

Create Backup

Load Backup: Select a backup file (WHX file) whose contents (either a file or disk sectors) you would like to restore.

Backup Manager

Execute: Executes the current file if exectuable, or otherweise the associated program.

Print

Properties: Lets you edit the size, the time stamp and attributes of a file (under Windows NT as well of a directory). Valid attributes are: A (archive), S (system), H (hidden), R (read-only). After entering new values in any area (size, time or attributes), simply press the Enter key, so the modifications take effect.

Open Special: This command is used open several files that meet special requirements at a time. Select a folder in which to open files. Subfolders are browsed optionally. You may specify a file mask (e. g. "w*.exe"). There is also a switch that permits opening only those files that contain a certain text or certain hex values. The standard search dialogs are displayed upon request for this purpose. If WinHex is not set up to work as a viewer or in-place editor (this can be done in the Tools menu), you may choose an edit mode.

Save Modified Files: All files which have been changed are written to the disk.

Save All Files: All files that have not been opened in view mode are written to the disk.

Exit: Use this command to end WinHex. You will be prompted to save any modifications to files and disks.

Edit Menu

Undo: Reverses the last modification, in case the corresponding undo option was activated.

Cut: Removes the current block from the file and puts it into the clipboard. The data following the block is pulled to the former block beginning.

Copy Block/All/Sector:
- Normal: Copies the current block/the entire file/the current sector into the clipboard. The contents of the clipboard can be pasted or written later.
- Into New File: Copies the data directly into a new file (not via the clipboard). For instance, this command can be used to recover a lost file from disk sectors.
- Hex Values: Copies the data as concatenated hex values.
- C Source: Copies the data as C-formatted source code into the clipboard.
- Editor Display: Copies the data as text, formatted as if it was displayed in the hex editor, i. e. with an offset, a hex and a text column.

Paste Clipboard: Inserts the clipboard contents at the current position of a file. The file data following this position is moved forward.

Write Clipboard: Copies the clipboard contents to the current file at the current position. The data at this position is *overwritten*. If the end of the file is encountered, the file size is increased so that the clipboard contents finds place.

Paste Clipboard Into New File: Creates a new file of the clipboard contents.

Empty Clipboard: This command is used to free the memory used by the clipboard.

Remove: Deletes the current block from the file. The data following the block is pulled to the former block beginning. The clipboard is not affected by this command. If the block is equally defined in all open files (i.e. it begins and ends at the same offsets), this command can even be applied to all open files at the same time.

Paste Zero Bytes: Use this command to insert zero bytes at the current position of a file.

Define Block: This function is accessible from the menu and the status bar. A dialog box lets you specify the desired block limits. This command can also be applied to all open files.

Select All: Defines the beginning and the end of the current file as its block limits.

Convert

Modify Data

Fill Block/File/Disk Sectors

Search Menu

Find Text: This command is used to search for a specified string of up to 50 characters in the current file, disk or RAM section (cf. Search Options).

Find Hex Values: This command is used to search for a sequence of up to 50 two-character hex values (cf. Search Options).

Replace Text: Use this command to replace occurrences of a specified string with another string (each of up to 50 characters), cf. Replace Options.

Replace Hex Values: Functions exactly as the Replace Text command, but is applied to a sequence of hex values (50 at max.), cf. Replace Options.

Combined Search: Provides a complex search mechanism. In the current and in a second file a common offset is searched, where either file contains the specified respective hex values.

Integer Value: Enter an integer (within the limits of the signed 64-bit integer data type). This function searches data in the current file, which can be interpreted as this integer.

Floating-Point Value: Enter a floating-point number (e. g. 12.34 = 0.1234 * 10^2 = 0.1234E2) and select a floating-point data type. This function searches data in the current file, which can be interpreted as this floating-point value.

Text Passages: Use this command to look for a sequence of letters (a-z, A-Z), digits (0-9) and/or punctuation marks. It is useful for instance if you intend to translate text passages hidden somewhere in a file with executable code.
Set the sensitivity of the search by specifying how long a character sequence must be to be recognized. Click "Tolerate Unicode characters" in order to force the algorithm to accept zero bytes between two characters.

Continue Global Search: This command is used to continue a global search operation (i.e. a search operation applied to all opened files) in the next file.

Continue Search: Lets you continue a search operation in the current file at the current position.

Position Menu

Go To Offset: Moves the current position to the specified offset. Normally this is done relative to the beginning of the file (offset 0). You can also move the cursor relative to the current position (forward or backward) or from the end of the file (backward). An offset can be specified in bytes (default), words (2 bytes) or doublewords (4 bytes). Press F11 to repeat the last position movement.

Go To Page/Sector: Browses to the specified page, sector, or cluster. Please note that the data area on FAT drives starts with cluster #2.

Move Block: Moves the current block *selection* (not the data within the block) forward or backward. Specify the distance in bytes. Press Alt+F11 to repeat the last block movement, press Shift+Alt+F11 to reverse the movement. This command may facilitate editing a file that consists of homogeneous records of a fixed length.

Go To...

Beginning Of File: Display the first page of the current file and moves the current position to offset 0.

End Of File: Displays the last page of the current file and moves the current position to the last byte (offset=file size-1).

Beginning Of Block: Moves the current position to the beginning of the current block.

End Of Block: Moves the current position to the end of the current block.

Mark Position: Marks the current position and thus enables you to find it again later.

Delete Marker: Removes the marker from the screen.

Go To Marker: Moves the current position to the marker set by Mark Position.

Position Manager

Window Menu

Window Manager: Displays all windows and provides "instant window switching" functionality. You may also close windows and save files.

Close All: Closes all windows and thus all open files, disks and RAM sections.

Close All Without Prompting: Closes all windows and thus all opened files and disks without giving you the opportunity to save your modifications.

Cascade/Tile: Arranges the windows in the aforementioned way.

Synchronize Scrolling: Synchronizes up to 4 tiled windows.

Synchronize And Compare: Synchronizes 2 windows and visually displays byte value differences.

Minimize All: Minimizes all windows.

Arrange Icons: This command arranges all minimized windows.

Tools Menu

Disk Editor

Inspect Cluster Chains: Available for FAT16 and FAT32 drives. WinHex traverses all cluster chains and thereby generates a drive map. This enables WinHex to specify each sector's or cluster's allocation (file, directory, file allocation table, unused). It is recommended to invoke this command again after file operations on a drive to keep the information displayed by WinHex up to date. Cf. security options.

List File/Directory Clusters: Available for FAT16, FAT32, and NTFS drives. On FAT16 and FAT32 drives, this command relies on a recent drive map as generated by the Inspect Cluster Chains command. WinHex searches clusters that are assigned to a file or directory you specify. The found clusters are listed in a separate window. You may click a list item to browse to that cluster number.

Set Disk Parameters: Using this command on a physical disk, you may override the number of cylinders, heads, and sectors per track as recognized by WinHex. This can be useful to access surplus sectors at the end of the disk under Windows NT/2000 (they should be automatically included under Windows 9x/Me), or to adjust the CHS coordinate system to your needs. Use this command on a logical drive to override the total number of clusters WinHex detects on that drive. This can prove useful when examining huge DVDs, which are detected as 2 GB media under Windows 9x.

File Retrieval: A simple file recovery function, that searches for files on any disk (or disk image file) that can be recognized by a certain file header (signature). The function can either extract files of a fixed size, or search for corresponding footers (which mark the end of the file). Header and footer must be provided in hexadecimal notation. A log file that tells the header and footer offsets is written to the output directory as well. The resulting files are named according to a pattern you must provide. Note that this function may not work at all in the case of fragmented files, where headers and footers do not occur alternately.

Clone Disk

If you want to clone a floppy disk and you only have *one* floppy drive, you may create a backup of the entire disk and restore it back to a different disk.

RAM Editor

Invoke Text Editor: Executes an external text editor (set in the General Options dialog). Opens the current file in that program. If you alter the file using the text editor, WinHex can adopt the changes afterwards.

Calculator: Runs the Windows calculator "calc.exe". Switching to scientific mode is highly recommended.

Hex Converter: Enables you to convert hexadecimal numbers into decimal numbers and vice versa. Simply type in the number and press ENTER.

Tables: Provides four conversion tables (cf. ANSI ASCII/IBM ASCII).

Analyze Block/File/Disk: Scans the data within the current block/the entire file/the entire disk and counts the occurrences of each byte value (0...255). The result is graphically displayed by proportional vertical lines. The number of occurrences and the percentage are displayed for each byte value when moving the mouse over the corresponding vertical line.
Use this command for instance to identify data of unknown type. Audio data, compressed data,

executable code etc. produce characteristic graphics. Use the system menu of the window to switch zero byte consideration on or off.
A standard checksum and the CRC32 of the selected data are also displayed.

Calculate Digest: Calculates the common 128-bit MD5 message digest of the entire current file, disks, or the currently selected block. Use the key combination SHIFT+F2 to calculate the 256-bit PSCHF instead.

Routine Manager

Apply Routine: Select a routine from the list to execute the automatic operations it was defined with.

Options Menu

General Options

Undo Options

Security & Safety Options

Data Interpreter Options

Use As Viewer: If you use WinHex simply for viewing files and you do not need the possibility to edit them, you can switch on this menu option. All files will be opened in view mode.

In-Place Editor: All files are opened in in-place mode, i. e. all changes are done directly in the original file.

Character set: Lets you switch between the ANSI-ASCII, IBM-ASCII, and EBCDIC character for display and keyboard input. You may also use Shift+F7. EBCDIC (originating from IBM mainframes) is currently not supported by the print command.

Text Display Only: Hides the hexadecimal data display and uses the full width of the editor window for the ASCII text display.

Hex Display Only: Hides the ASCII text display and uses the full width of the editor window for the hexadecimal data display.

File Manager

Execute: Lets you select a file to execute. If the file itself is not executable, it is opened by the application it is associated with.

Split: This command creates several destination files using the contents of a single source file. Specify a split offset for each destination file. The source file is not affected by this function.

Concatenate: Select several source files that are to be copied into one destination file. The source files are not affected.

Unify: Select two source files and one destination file. The bytes/words from the source files will be written alternately into the destination file. The first byte/word originates from the source file that was specified first. Use this function to create a file with odd and even bytes/words originating from separate files (e.g. in EPROM programming).

Dissect: Select a source file and two destination files. The bytes/words from the source files will be written alternately into the destination files. The first byte/word will be transfered to the destination file that was specified first. Use this function to create two separate files each containing either the odd or the even bytes/words of the original file (e.g. in EPROM programming).

Compare: This command is used to compare two files byte by byte. Decide whether different or identical bytes shall be reported. If desired, the operation terminates itself when having found a certain number of differences or identical bytes. The report is stored as a text file, whose size might otherwise grow dramatically.
Optionally, the command can be applied to a limited block in the files. WinHex automatically specifies the end of the shorter file as the end of the block.

Copy: Simply copies a file.

Move: Use this command to move and/or rename an existing file. The source file is deleted.

Delete Irreversibly: This command is used to erase the contents of a file irrevocably, so that it cannot be restored by special undelete programs. Each selected file is filled with zero bytes, shortened to a length of zero and then deleted. The name entry of the file is erased as well. Even professional attempts to restore the file will be futile. Therefore this command should be applied to files with confidential contents, which is to be destroyed. However, the *absolute* effectiveness of this function cannot be guaranteed, especially when conflicting with resident security utilities, which use elaborate mechanisms to maintain data integrity.

Help Menu

Contents: Displays the contents of this help file.

Setup: Lets you switch between the English, the German, the French, and the Spanish user interface.

Initialize: Use this command to restore the default settings of this program.

Uninstall: Use this command to remove WinHex from your system. This works properly even if you did not install WinHex using the setup program.

Homepage: Opens the WinHex homepage (http://www.winhex.com) in your browser.

Printing

Use the "print" command of the File menu to print a file, disk sectors or RAM contents. Define the printing rang via offsets. You may select and set up a printer.

Please choose the character set for printing and accept or change the suggested font size. The recommended font size is calculated as follows: print resolution (e. g. 720 dpi) / 6 (e. g. = 120). If desired you may enter a comment which will be printed at the end.

If you need more flexibility with printing, you can define a block and copy it using "Edit->Copy->Editor Display" as a hex-editor-formatted text into the clipboard. You may paste it in your favorite word processor. It should look perfect in "Courier New", 10 pt.

Block

You can mark a part of an open file as a "block". This part can be manipulated by several function in the edit menu just as selections in other Windows programs. If no block is defined, these functions usually are applied to the whole file.

The current position and size of the block are displayed in the status bar. Double-clicking the right mouse button or pressing the ESC key clears the block.

Modify Data

Use this command to modify the data within the block or within the whole file, in case no block is defined. Either a fixed integer number is *added* to each element of the data, the bits are *inverted*, a constant is XORed with the data (a simple kind of encryption), bits are shifted logically, or bytes are swapped. By shifting bits, you can simulate inserting or removing single bits at the beginning of the block.

Swap Bytes

This command assumes all data to consist of 16-bit elements (32-bit elements resp.) and swaps high-order and low-order bytes (and high-order and low-order words resp.). Use it in order to convert big-endian into little-endian data and vice versa.

Addition

Specify a positive or negative, decimal or hexadecimal number, which is to be added to each element of the current block. An integer format defines size (1, 2 or 4 bytes) and type (signed or unsigned) of an element.

There are two ways how to proceed if the result of the addition is out of the range of the selected integer format. Either the range limit is assumed to be the new value (**I**) or the carry is ignored (**II**).

Example: unsigned 8-bit format

I.	FF + 1 -> FF	(255 + 1 -> 255)
II.	FF + 1 -> 00	(255 + 1 -> 0)

Example: signed 8-bit format

I. 80 - 1 -> 80 (-128 - 1 -> -128)
II. 80 - 1 -> 7F (-128 - 1 -> +127)

• If you decide to use the first method, WinHex will tell you how often the range limit has been exceeded.

• The second method makes sure the operation is reversible. Simply add *-x* instead of *x* based on the same integer format to recreate the original data.

• When using the second method it does not make a difference whether you choose a signed or an unsigned format.

Conversions

WinHex provides the "Convert" command of the Edit menu for easy conversions of different data formats and for encryption and decryption. The conversion can optionally be applied to all opened files instead of only the currently displayed one. The formats marked with an asterisk can only be converted as a whole file, not as a block. The following formats are supported:

• ANSI ASCII, IBM ASCII (two different ASCII character sets)
• EBCDIC (an IBM mainframe character set)
• Lowercase/uppercase characters (ANSI ASCII)
• Binary* (raw data)
• Hex ASCII* (hexadecimal representation of raw data as ASCII text)
• Intel Hex* (=Extended Intellec; hex ASCII data in a special format, incl. checksums etc.)
• Motorola S* (=Extended Exorcisor; ditto)

Please note:
• When converting Intel Hex or Motorola S data, the internal checksums of these formats are not checked.
• Depending on the file size, the smallest possible output subformat is chosen automatically. Intel Hex: 20-bit or 32-bit. Motorola S: S1, S2, or S3.
• Some conversions can only be applied to whole files.

Encryption/Decryption

It is recommended to specify a combination of at least 8 characters as the encryption key. Do not use words of any language, it is better to choose a random combination of letters, punctuation marks, and digits. Note that encryption keys are case sensitive. Remember that you will be unable to retrieve the encrypted data without the appropriate key. The decryption key you enter is not verified before decrypting.

The encryption algorithm is "Pukall Cipher 1" (PC 1), using a 128-bit key (=the 128-bit digest of the key you specify).

Search Options

Case sensitive: If this option is enabled, WinHex distinguishes between upper and lower case, e.g. so that "Option" is not found in the word "optionally".

Unicode character set: The specified text is searched using the 256 ANSI-ASCII-equivalent Unicode characters, where the high-order byte is 0.

You may specify a **wildcard** (one character or a two-digit hex value), which represents one byte. For example this option can be used to find "Speck" as well as "Spock" when searching for "Sp?ck" with the question mark as the wildcard.

Only whole words: The searched string is recognized only, if it is separated from other words, e. g. by punctuation marks or blanks. If this option is enabled, "tomato" is not found in "automaton".

Search direction: Decide whether WinHex shall search from the beginning to the end, or downwards or upwards from the current position.

Condition: Offset modulo x = y: The search algorithm accepts search string occurrences only at offsets that meet the given requirements. E. g. if you search for data that typically occurs at the 10th byte of a hard disk sector, you may specify x=512, y=10. If you are looking for DWORD-aligned data, you may use x=4, y=0 to narrow down the number of hits.

Search in block only: The search operation is limited to the current block.

Search in all opened files: The search operation is applied to all opened files. Press F4 to continue the search in the next file. If "Search in block only" is enabled at the same time, the search operation is limited to the current block of each file.

Count occurrences/Save occurrence positions: Forces WinHex not to show each single occurrence, but to count them. If this option is fully enabled, WinHex will enter all occurrences into the Position Manager.

Search Menu
Replace Options
Technical Hints

Replace Options

Prompt when found: WinHex awaits your decision when an occurrence has been found. You may either replace it, continue or abort the search.

Replace all occurrences: All occurrences are replaced automatically.

Case sensitive: The characters that are to be replaced are searched using this option (cf. Search Options).

Unicode character set: The specified characters are searched and replaced in Unicode format (cf. Search Options).

You may specify one character/a two-digit hex value as a **wildcard** (cf. Search Options). This is usually done in the search string.
If the *substitute* contains a wildcard, the character at the relative position in an occurrence will not be changed. Thus, "black" and "block" can be replaced simultaneously with "crack" and "crock" (enter "bl?ck" and "cr?ck").

Only whole words: The searched string is recognized only if it is separated from other words e. g. by punctuation marks or blanks. If this option is enabled, "tomato" is not replaced in "automaton".

Search direction: Decide whether WinHex shall replace from the beginning to the end, or downwards or upwards from the current position.

Replace in block only: The replace operation is limited to the current block.

Replace in all opened files: The replace operation is applied to all files not opened in view mode. If "Replace in block only" is enabled at the same time, the replace operation is limited to the current block of each file.

WinHex is able to replace one string or hex value sequence with another one that has a different length. You will be prompted, which of the following methods shall be applied:

1st method: The data behind the occurrence is moved due to length difference. So the file size is changed. This method must not be applied to certain file types, such as executable files. It is even possible to specify *nothing* as the substitute, which means all occurrences will be *removed* from the file!

2nd method: The substitute is written into the file at the position of the occurrence. If the substitute is shorter than the searched character sequence, the exceeding characters will remain in the file. Otherwise even the bytes behind the occurrence will be overwritten (as far as the end of the file is not reached). The file size is not affected.

Search Menu
Search Options
Technical Hints

Undo Options

The availability of the "Undo" command depends on the following options:

• Specify how many sequential actions are to be reversed by the Undo command. This option does not affect the number of reversible keyboard inputs, which is only limited by the available RAM.

• In order to save time and space on your hard disk, you can specify a file size limit. If a file is larger than this limit, backups will not be created and the Undo command is not available except for keyboard input.

• Automatically created backups for the internal use with the Undo command are deleted by WinHex when closing the file, if the corresponding option is fully selected If it is partially selected, they are deleted when WinHex terminates.

• Choose for all kinds of editing actions whether they should be reversible or not. In case they should, an internal backup is created before the action takes place.

Position Manager

The Position Manager maintains a list of file or disk offsets and corresponding descriptions. You may enter new positions and edit or delete existing entries. If a special offset in a file is important to you because you have to edit it more than once, you can enter it into the Position Manager. This makes it a lot easier to find it again later, and you do not have to remember it. An appropriate description for instance could be "Data chunk begins here!".

Click the right mouse button in order to see a context menu. The context menu provides additional commands. You may delete, load or save positions, even export the list as HTML. If the position list was changed, it is saved in the file *WinHex.pos* when exiting WinHex.

The position manager window can be minimized, so you may switch between the positions in the selected order by pressing Ctrl+Left and Ctrl+Right.

A complete documentation of the POS file format is available from the WinHex Homepage http://www.winhex.com.

Routine Manager

The Routine Manager is a multifunctional dialog window which displays the existing routines. The list can be sorted by the routine number, the routine title or the files the routines apply to. You can edit, copy and delete existing routines and create new ones.

When deleting a routine, the numbers of the following routines are decreased by 1. References to those routines as "next routines" are updated.

Backup Manager

Displays a list of previously created backups. The items can be listed in a chronological or alphabetical order. Choose the backup you would like to restore. When the function completed, the original file or sector contents is shown. It is not written to the disk until you use the Save command. Disk sectors can optionally be written *immediately* to the disk or into a new file. You may also wish to specify a different destination disk or a different destination sector number. It is also possible to only extract a subset of the sectors from the backup. (However, sectors at the beginning of a *compressed* backup cannot be left out during restoration.)

If the backup was saved with a checksum and/or a digest, data authenticity is verified before the sectors will be directly written to the disk.

The backup manager also allows to delete backups, which you do not need any longer. Backups that were created for internal use by the Undo command can be deleted by WinHex automatically (cf. Undo Options).

Backup files that are maintained by the Backup Manager are located in the folder specified in the General Options dialog. Their filenames are "Savedxxx.whx" where xxx is a unique three-digit identification number. This number is displayed in the last column of the backup manager list.

A complete documentation of the WHX file format is available from the WinHex Homepage http://www.winhex.com.

Data Interpreter

The Data Interpreter is a small window that provides "translation services" for the data at the current cursor position. The Data Interpreter Options dialog allows you to hide or show the Data Interpreter and lets you specify the data types to interpret. These are currently seven integer data types, the binary format (8 bits of a byte), four floating-point data types, assembler opcodes (Intel), and five date types.

The Data Interpreter is also capable of translating all data types (except assembler opcodes) back into hex values. Double-click a number in the Data Interpreter window, enter a new value and press ENTER. The Data Interpreter will enter the corresponding hex values into the edit window at the current position.

Right-click the data interpreter to bring up a context menu. This will let you switch between big-endian and little-endian translation of integer and floating-point data.

Hints:

Some hex values cannot be translated into floating-point numbers. For these hex values the Data Interpreter displays NAN (**n**ot **a** **n**umber).

Some hex values cannot be translated into valid dates. The value ranges of different date types are more or less narrow.

There are redundancies in the Intel instruction set, which show up in the Data Interpreter as duplication of both hex opcodes and mnemonics. Floating-point instructions are generally displayed as F***.

More detailed reference can be found in the Intel Architecture Software Developer’s Manual Volume 2: Instruction Set Reference, available in PDF format on the Internet.

RAM Editor

The RAM Editor is part of the Tools menu. It allows examining the virtual memory of a process (i.e. a program that is being executed). All memory pages committed by this process are presented in a continuous block. Unused (free or reserved) pages are ignored by WinHex.

Select one of the listed processes. You may access either the so-called primary memory or the entire memory of this process or one of the loaded modules. The primary memory is used by programs for nearly all purposes. Usually it also contains the main module of a process (the EXE file), the stack, and the heap. The "entire memory" contains the whole virtual memory of a process, including the part of memory that is share among all processes, except system modules. Under Windows 95/98, system modules are listed optionally in the process tree. System modules are defined as modules that are loaded above the 2 GB barrier (such as kernel32.dll, gdi32.dll). They are shared among all running processes.

Please note the following limitations:

• Caution: Only keyboard input can be undone!
• Virtual memory of 16-bit processes is *partially* accessible under Windows 95/98 only.
• Editing is possible in in-place mode only.
• System modules of Windows 95/98 can only be *examined* in view mode, *not manipulated*.
• The evaluation version only supports view mode. Order the full version!

The options relevant for the RAM editor are "Check for virtual memory alteration" (security options) and "Virtual addresses" (general options).

Context Menu

The Windows shell displays the context menu when the user clicks an object with the right mouse button. WinHex is present in the context menu only if you enable to corresponding option.

Edit with WinHex: Opens the selected file in WinHex.

Open Folder in WinHex: Lets you open all files of the selected folder in WinHex, just as the Open Special command of the File menu.

Edit Disk: Opens the selected disk in the disk editor of WinHex. If you hold the Shift key, instead of the selected logical drive the corresponding physical disk is opened, if any. (The latter feature is not available under Windows NT.)

WinHex provides its own context menus on the status bar, the Data Interpreter, and in the position manager.

Your license codes were probably valid for *earlier* versions of WinHex. You cannot run *this* version as a full version using these codes. Please see http://www.winhex.com/winhex/upgrade.html for information about upgrades. If you paid for WinHex 9.0 or later, upgrading to this version is *free* for you (send e-mail to mail@sf-soft.com and specify your name and address).

Key

Specify a string consisting of 1-16 characters as the encryption/decryption key. The more characters you enter, the safer is the encryption. The key itself is not used for encryption and decryption, instead it is digested to the actual key.

Your key is not saved on your hard disk. If the corresponding security option is enabled, the encrypted key is stored in the RAM, as long as WinHex is running.

Create Backup

This command provides a dialog window enabling you to create a backup (security copy) of the current file or of sectors of the current disk (drive imaging). The backup is stored as a WHX file.

When creating a backup of a physical disk or logical drive (so-called drive imaging), you have to specify which sectors to save. By default, all sectors starting with the current position are selected. You may have WinHex split the backup into volumes of a fixed size. Partial backups of 650 MB e.g. are suitable for archiving drive images on CD-R. In order to make the backup as small as possible, it is recommended to initialize unused space before drive imaging. This is because sectors that consist but of zero values do not increase the backup size when compression is enabled.

Hint on disk cloning and disk imaging

If you have WinHex assign a filename for the WHX file automatically, the file will be created in the folder for backups (cf. General Options) and will be available in the Backup Manager. If you explicitly specify a path and a filename, you can restore the backup later using the »Load Backup« command.

You may specify a textual description of the backup.

The WHX format is able to store a checksum (CRC32) and a digest of the original data, and it optionally provides encryption. The calculation of digests and encryption both slow down the backup creation considerably, so these functions should not be used unless required for security purposes. When restoring a digested backup with no warning showing up in WinHex, you can be sure about the data contained in the backup has not been tampered with!

The encryption algorithm is "Pukall Cipher 1" (PC 1), using a 128-bit key that is digested from the 256-bit concatenation of the 128-bit digest of the key you enter and 128 bits random input. The random input is saved in the WHX file for later decryption.

WinHex makes use of the "Deflate" compression algorithm that is part of the popular general-purpose library *zlib*. This algorithm consists of LZ77 compression and Huffman coding. The compression ration is the same as ZIP.

The WHX file format is fully documented (cf. the WinHex Homepage http://www.winhex.com).

Security & Safety Options

• Use the option **Restrictive drive control** to make sure the disk editor locks a disk for other programs before accessing it, even if it is not really necessary. This may result in a loss of speed.

• The **Sector reading cache** accelerates sequential disk access by the disk editor. This option is recommended particularly when scrolling through CD-ROM and floppy disk sectors, since the number of necessary physical accesses is significantly reduced.

• Enabling the option **Inspect clusters automatically** causes WinHex to automatically traverse the cluster chains of a FAT16 or FAT32 drive if such a drive is opened in WinHex and the required drive map does not yet exist. Using the drive map, WinHex is able to display each sector's and cluster's allocation (for storing which file it is used). Use the command "Inspect clusters" of the Tools menu to update the drive map.

• With the option **Keep drive map for next session** enabled, all information on FAT16 and FAT32 drives collected by WinHex remains in the folder for temporary files even when WinHex terminates. WinHex can reuse drive maps during later program runs.

• Use the option **Check for virtual memory alteration** to make sure the RAM editor inspects the structure of virtual memory every time before *reading* from or *writing* to it. If the structure has changed, a possible read error is prevented. Especially under Windows NT the checking may result in a loss of speed. When editing the "entire memory" of a process, WinHex generally *never* checks for alterations, even if this option is enabled.

• A checksum can be **calculated** for each file when opening it. It is then displayed in the information column at the right. **Checksums** can also be calculated by analyzing the current block (cf. Tools Menu).

• Before modifications to an existing file are saved (i. e. before the **file** is **updated**), you are prompted for **confirmation**. To inhibit this behavior of WinHex, switch off the corresponding option.

• When manually restoring backups, a **restoration report** is **shown** only in case the backup contains a digest or the backup is corrupt. Optionally, you may have WinHex always display the report **always** after restoration. In this case even the digest will be displayed.

• The **key** that is required for encryption and decryption can be entered in a normal edit box. Optionally, you **enter** it **blindly** (asterisks are displayed instead of the actual characters). In this case you have to confirm the key in a second edit box to detect typos.

• By default, the encryption **key** is kept in main memory (in an encrypted state) as long as WinHex is running so that you do not have to type it again and again if you use it several times. Possibly you prefer WinHex to erase the key after use.

Endian-ness

Microprocessors differ in the position of the least significant byte: Intel® and MIPS® processors have the least significant byte first. A multi-byte value is stored in memory from the lowest byte (the "little end") to the highest byte. For example, the hexadecimal value 12345678 is stored as 78 56 34 12. This is called the **little-endian format**.

Motorola processors have the least significant byte last. A multi-byte value is stored in memory from the highest byte (the "big end") to the lowest byte. For example, the hexadecimal value 12345678 is stored as 12 34 56 78.   This is called the **big-endian format**.

Digests

A so-called digest is, similar to a checksum, a characteristic number used for verification of data authenticity. But digests are more than that: digests are strong one-way hash codes.

It is computationally feasible to manipulate any data in such a way that its checksum remains unaffected. Verifying the checksum in such a case would lead to the assumption that the data has not been changed, although it has. Therefore, digests are used instead of checksums if malicious (i.e. not mere random) modifications to the original data are to be detected. It is computationally infeasible to find any data that corresponds to a given digest. It is even computationally infeasible to find two pieces of data that correspond to the same digest.

Of course, random modifications, e. g. caused by an inaccurate transmission, can also be detected when using digests, but checksums serve better for this purpose, because they can be calculated much faster.

WinHex uses 128-bit digests for encryption key generation (cf. Conversions and Create Backup) and 256-bit digests for data authenticity verification in backup files. The "Pukall Stream Cipher" hash function with a fixed 128-bit key (F6 C7 24 95 17 9F 3F 03 C6 DE F1 56 F8 2A 85 38) is used in WinHex to calculate digests.

WinHex also incorporates the 128-bit MD5 message digest.

Date Types

The following date formats are supported by the Data Interpreter:

MS-DOS Date & Time (4 bytes)

The lower word determines the time, the upper word the date. Used by several DOS function calls and by all FAT file systems.

Bits	Contents
0-4	Second divided by 2
5-10	Minute (0-59)
11-15	Hour (0-23 on a 24-hour clock)
16-20	Day of the month (1-31)
21-24	Month (1 = January, 2 = February, etc.)
25-31	Year offset from 1980

Win32 FILETIME (8 bytes)

The FILETIME structure is a 64-bit integer value representing the number of 100-nanosecond intervals since January 1, 1601. Used by the Win32 API.

OLE 2.0 Date & Time (8 bytes)

A floating-point value (more exactly: a double) whose integral part determines the number of days passed since December 30, 1899. The fractional part is interpreted as the day time (e.g. 1/4 = 6:00 a.m.). This is the OLE 2.0 standard date type, e.g. it is used by MS Excel.

ANSI SQL Date & Time (8 bytes)

Two consecutive 32-bit integer values. The first one determines the number of days since November 17, 1858. The second one is the number of 100-microsecond intervals since midnight. This is the ANSI SQL standard and used in many databases (e.g. InterBase 6.0).

UNIX/C Date & Time (4 bytes)

A 32-bit integer value that determines the number of seconds since 1/1/1970. This data type is used in UNIX, DOS C and C++ ("time_t"), and by FORTRAN programs since the 80's. Sporadically defined as the number of *minutes* since 1/1/1970. The Data Interpreter options let you switch between both sub-types.

Master Boot Record

The **Master Boot Record** is located at the physical beginning of a hard disk, editable using the Disk Editor. It consists of a **master bootstrap loader code** (446 bytes) and four subsequent, identically structured **partition records**. Finally, the hexadecimal signature 55AA completes a valid Master Boot Record.

The format of a partition record is as follows:

Offset	Size	Description
0	8 bit	A value of 80 designates an active partition.
1	8 bit	Partition start head
2	8 bit	Partition start sector (bits 0-5)
3	8 bit	Partition start track (bits 8,9 in bits 6,7 of sector)
4	8 bit	Operating system indicator
5	8 bit	Partition end head
6	8 bit	Partition end sector (bits 0-5)
7	8 bit	Partition end track (bits 8,9 in bits 6,7 of sector)
8	32 bit	Sectors preceding partition
C	32 bit	Length of partition in sectors

Operating system indicators: (hexadecimal)

00	Empty partition-table entry
01	DOS 12-bit FAT
04	DOS 16-bit FAT (up to 32M)
05	DOS 3.3+ extended partition
06	DOS 3.31+ Large File System (16-bit FAT, over 32M)
07	OS/2 HPFS, Windows NT NTFS, Advanced Unix
08	OS/2 v1.0-1.3, AIX bootable partition, SplitDrive
09	AIX data partition
0A	OS/2 Boot Manager
0B	Windows 95 with 32-bit FAT
0C	Windows 95 with 32-bit FAT (using LBA-mode INT 13 extensions)
0E	Logical-block-addressable VFAT (same as 06 but using LBA-mode INT 13)
0F	Logical-block-addressable VFAT (same as 05 but using LBA-mode INT 13)
17	Hidden NTFS partition
1B	Hidden Windows 95 FAT32 partition
1C	Hidden Windows 95 FAT32 partition (using LBA-mode INT 13 extensions)
1E	Hidden LBA VFAT partition
50	OnTrack Disk Manager, read-only partition
51	OnTrack Disk Manager, read/write partition
81	Linux
82	Linux Swap partition, Solaris (Unix)
83	Linux native file system (ext2fs/xiafs)
85	Linux EXT
86	FAT 16 volume/stripe set (Windows NT)
87	HPFS fault-tolerant mirrored partition, NTFS volume/stripe set
BE	Solaris boot partition
C0	DR-DOS/Novell DOS secured partition
C6	Corrupted FAT 16 volume/stripe set (Windows NT)
C7	Corrupted NTFS volume/stripe set
F2	DOS 3.3+ secondary partition

Fill Block/File/Disk Sectors

Fill with hex values: Specify up to 5 two-character hex values, which will be copied repeatedly into the current block, the entire file or all disk sectors, respectively.

Fill with random bytes: Specify a decimal interval (0 to 255 at max.) for random numbers, which will be copied repeatedly into the current block, the entire file or all disk sectors, respectively. The random bytes are Laplace-distributed.

Generate chaotic numbers: Generates a series of chaotic numbers from a (random) initial fraction using a Mixmaster algorithm. The numbers will be copied into the current block, the entire file or all disk sectors, respectively. The initial fraction, that you may define yourself, must be of the form "x.y", where x is <256 and y consists of at least one digit.

In case in *all* open files either a *block* or *no* block is defined, this command can optionally be applied to all these files at the same time.

Disk Editor: Questions and Answers

How can I access CD-ROM sectors under Windows 9x?

Please make sure the following requirements are met:

1. A Windows driver of the CD-ROM drive must be installed. An MS-DOS driver is not sufficient.

2. The ASPI interface must be installed. Maybe you have to copy the file wnaspi32.dll manually into your Windows\System directory. The file is to be found on your Windows installation CD. The shareware program WinZip (available from http://www.winzip.com) is recommended for extracting files from CAB archives.

3. The CD-ROM drive must support the way WinHex tries to read sectors. Most of modern ATAPI and SCSI drives are suitable.

How can I make WinHex detect an installed PC Card ATA Flash Disk/PCMCIA Drive as a physical disk under Windows 9x?

Windows Control Panel -> System -> Device Manager -> Select your PCMCIA drive -> Click "Properties" and search for an option with name similar to "Int 13h device". The actual way to find this checkbox may vary on different Windows versions. If possible, *enable* this option and reboot your computer.

Template Editing

A template is a dialog box that provides means for editing custom data structures in a more comfortable and error-preventing way than raw hex editing does. Editing is done is separate edit boxes. Changes take effect when pressing the Enter key or when quitting the template after being prompted. The data may originate from a file, from disk sectors, or from virtual memory. Especially when editing databases, you may prefer to define a custom template for ease of access to the records.

A template definition is stored in a text file. The template editor enables you to write template definitions and offers syntax checking. A template definition mainly contains variable declarations, that are similar to those in source code of programming languages. The supported data types include all the common integer, floating-point and boolean variants, five date types, hex values, binary, characters, and strings type. Arrays of both single variables and groups of variables can be used. The ability to move freely forwards and backwards within the data makes using templates particularly flexible:
• The same variable may be interpreted and manipulated in several ways.
• Irrelevant data sections can be skipped.

The template manager lists all text files in the WinHex directory that contain template definitions. The title of the template along with a description, the filename, and the date and time of the last modification is shown. Click the Apply button to display a template using the selected template definition for the data in the current editor window at the current position. You may also create a new template definition, delete or edit an existing one.

WinHex comes with several demonstration templates.

Template Definition

A template definition consists of a header and a body.

Header syntax
Variable declaractions in the body
Advanced commands the body

Template Definition Header

The header of a template definition has the following format:

template "*title*"
[description "*description*"]
[appliesto (file/disk/RAM)]
[sector-aligned]
[requires *offset* "*hex values*"]
[big-endian]
[hexadecimal]
[read-only]
[multiple [*fixed overall size*]]
// Put any general comments to the template here.
begin
    variable declarations
end

Tags in brackets are optional. The order of the tags is irrelevant. Expressions need only be enclosed in inverted commas if they contain space characters. Comments may appear anywhere in a template definition. Characters following a double slash are ignored by the parser.

The keyword "appliesto" must be followed by one and only one of the words file, disk, or RAM. WinHex issues a warning if you are going to use a template on data from a different source.

If the template applies to a disk, the keyword "sector-aligned" ensures the template interpretation starts at the beginning of the current sector, regardless of the exact cursor position.

Similar to the "appliesto" statement, the "requires" statement enables WinHex to prevent an erroneous application of a template definition to data that does not fit. Specify an offset and a hex-value chain of an arbitrary length that identifies the data for which the template definition was intended. For example, a valid master boot record can be recognized by the hex values 55 AA at offset 0x1FE, an executable file by the hex values 4D 5A ("MZ") at offset 0x0. There may be multiple "appliesto" statements in a template definition header, that are all considered.

The keyword "big-endian" causes all multi-byte integer and boolean variables in the template definition to be read and written in big-endian order (high-order byte first).

The keyword "hexadecimal" causes all integer variables in the template definition to be displayed in hexadecimal notation.

The keyword "read-only" ensures that the template can only be used to examine, but not to manipulate data structures. The edit controls within the template will be grayed out.

If the keyword "multiple" is specified in the header, WinHex allows browsing to neighboring data records while displaying the template. This requires that WinHex has knowledge of the record's size. If it is not specified as a parameter to the "multiple" statement, WinHex assumes the overall size of a template structure (=record) to be the current position at the end of the template interpretation less the base editing position. If this is a variable size, i.e. array sizes or move parameters are determined dynamically by the value of variables, WinHex cannot browse to precedent data records.

Variable Declarations

The body of a template definition mainly consists of variable declarations, similar to those in programming languages. A declaration has the basic form

type "title"

where type can be one of the following:

- int8, uint8 = byte, int16, uint16, int32, uint32, int64,
- binary,
- float = single, real, double, longdouble = extended,
- char, char16, string, string16,
- boole8 = boolean, boole16, boole32
- hex,
- DOSDateTime, FileTime, OLEDateTime, SQLDateTime, UNIXDateTime = time_t

"title" must only be enclosed in inverted commas if it contains space characters. "title" must not consist only of digits. WinHex does not distinguish between upper and lower case characters in titles. 41 characters are used to identify a variable at most.

"title" can be preceded by at most one member of each of the following modifier pairs:

big-endian	little-endian	(see Endian-ness)
hexadecimal	decimal	
read-only	read-write	

These modifiers only affect the immediately following variable. They are redundant if they appear in the header already.

The number at the end of a type name denotes the size of each variable (strings: of each character) in bits. With char16 and string16, WinHex supports Unicode characters and strings. However, Unicode characters other than the first 256 ANSI-equivalent characters are not supported. The maximum string size that can be edited using a template is 8192 bytes.

The types string, string16, and hex require an additional parameter that specifies the number of elements. This parameter may be a constant or a previously declared variable. If it is a constant, it may be specified in hexadecimal format, that is recognized if the number is preceded by 0x.

You may declare arrays of variables by placing the array size in square brackets next to the type or the title. The following two lines declare a dynamically sized ASCII string, whose length depends on the preceding variable:

```
    uint8                  "len"
    char[len]      "A string"
```

The same could be achieved by the following two declarations:

```
    byte                   "len"
    string len     "A string"
```

The character "~" can be used as a placeholder for later replacement with the actual array element number (see Advanced Commands). This does not apply to arrays of char variables, since they are automatically translated into a string.

Please note that in the current version templates are not able to "calculate", so operators such as "+" and "*" must not be used neither in parameters nor in array size expressions.

Advanced Commands

When enclosed in braces, several variable declaractions comprise a block that can be used repeatedly as a whole. Note, however, that blocks must not be *nested* in the current implementation. The "~" character can be used in a variable's name as a placeholder for later replacement with the actual repetition count. The optional "numbering" statement defines where to begin counting (0 by default).

```
numbering   1
{
byte                    "len"
string len     "String No. ~"
} [10]
```

In this example the actual variable names in the template will be "String No. 1", "String No. 2", ..., "String No. 10".

In order to facilitate reading and navigating the template, you may define groups of variables that are separated by empty space in the dialog box:

```
section        "...Section Title..."
...variable declaractions...
endsection
```

The "section", "endsection", and "numbering" statements do not advance the current position in the data to be interpreted.

There are two commands that do not declare variables either, but are explicitly used to change the current position. This can be done to skip irrelevant data (forward movement) or to be able access certain variables more than once as different types (backward movement). Use the "move n" statement to skip n bytes from the current position, where n may be negative. "goto n" browses to the specified absolute position from the beginning of the template interpretation (must be positive).

The following example demonstrates how to access a variable both as a 32-bit integer and as a four-part chain of hex values:

```
int32                   "Disk serial number (decimal)"
move -4
hex 4                   "Disk serial number (hex)"
```

Disk Cloning

The command "Clone Disk" is part of the Tools menu. This function copies a defined number of sectors from a source to a destination disk. Both disks must have the same sector size. In order to effectively *duplicate* a drive (i.e. in order to copy all sectors of the drive), enable the appropriate option, so the correct number of sectors is entered automatically. The destination disk must not be smaller than the source disk.

Disk cloning offers options that control the behavior when bad sectors are encountered on the source disk:
• By default, you are notified of the error and prompted for either continuing or aborting the operation. "Log procedure silently" creates a complete log file of the entire operation, including a report on unreadable sectors, and prevents WinHex from reporting each unreadable sector separately. This may prove useful e.g. for computer forensics.
• WinHex can either leave the destination sector that corresponds to a damaged source sector unchanged or fill it with zero bytes.

Regular disk cloning is not an option if you want to duplicate a disk in a removable drive (e.g. a floppy disk) with only one removable drive present. The correct concept for this application is *disk imaging* (could also be called "delayed" disk cloning). The image can be restored to a different disk. The result is the same as disk cloning.

There are two ways to image a disk:
• If convenience is your primary concern, use the backup functionality. For easy recovery, a backup file includes information on its contents: sector numbers, source disk etc.
• The disk cloning dialog allows copying sectors from a disk into a raw, headerless image file and later vice-versa. Combined with the silent "log file" mode, this is preferable to creating a backup in case of defective sectors on the source disk.

Hint on disk cloning and disk imaging

Cloning or imaging the drive that contains the active Windows installation can produce inconsistent copies. At any rate, please make sure the source drive is not written to during the cloning or imaging procedure by any other program or by Windows itself. It is recommended to move the TEMP directory to a different drive. The swap file should be created on a different drive, too. Alternatively, complete disabling swapping in the control panel may help as well.