PENG 2.1

A Powerful
And Portable
Text Editor

Frank Heckenbach

15 Acknowledgements

For beta-testing, bug reports and useful feature suggestions (in alphabetical order) to:

- Anja Drewitz ⟨`anja(at)theo-phys.uni-essen.de`⟩ – also for setting up the mailing list
- Dominik Freche ⟨`dominik.freche(at)mailbox.tu-dresden.de`⟩
- Jan Andres ⟨`jan(at)dinet.de`⟩
- Nicola Girardi ⟨`girardi(at)keycomm.it`⟩
- Peter Gerwinski ⟨`peter(at)gerwinski.de`⟩

To all the developers of GPC, GCC, ncurses, PDCurses, rx, Linux, DJGPP, TeX, the Texinfo tools and the GNU project for writing all these programs and libraries without which this program would not have been possible.

The source and binaries of the ncurses, PDCurses (both including panels) and rx libraries are available in ‘ftp://agnes.dida.physik.uni-essen.de/gnu-pascal/libs/’ and subdirectories. Object files of the program, including binaries of the GNU Pascal runtime library, should be available on the main distribution place of the program (if not, write to the author). Note that the GNU Pascal runtime library is changing fast, so future versions of PENG will probably be linked to future versions of that library which will then be provided.

The DJGPP version of the program needs a slightly modified version of DJGPP 2.02. If you want to relink the DJGPP version of the programs, you need ‘djlsr202.zip’ from the DJGPP distribution and the patch provided at ‘ftp://agnes.dida.physik.uni-essen.de/gnu-pascal/djgpp/dj202dif.zip’. If you just want to use the PENG binary, you do not need this patch.

The object files and libraries can be linked to form an executable e.g., by moving the libraries into the usual lib directories and doing:

- ‘gcc -s -o peng peng.a libgpc.a -lm -lpanel -lncurses -lrx’ (Unix, terminal) or
- ‘gcc -s -o peng peng.a libgpc.a -lm -lXPanel -lXCurses -lXaw -lXmu -lXt -lX11 -lSM -lICE -lXext -lrx’ (Unix, X11) or
- ‘gcc -s -o peng peng.a libgpc.a -lm -lpanel -lcurso -lrx’ (DJGPP).

Under DJGPP, you should increase the stack size of the program by doing ‘stubedit peng.exe minstack=2m’ after linking, otherwise you might experience mysterious crashes or even system hangs.

Table of Contents

14 Legal Stuff

The main distribution place of the program is currently 'http://fjf.gnu.de/peng.html'.

You are not required to accept the following license terms, since you have not signed them. However, nothing else grants you permission to use the program. Therefore, if you don't agree to these terms, you cannot use the program.

You can terminate this license at any time. If you fail to comply to any term of this license, your license to use the program will terminate immediately automatically.

The program may be used for private purposes only. Any commercial use requires a special license from the author. Modification of the program for your own use and reverse engineering for debugging such modifications is allowed. Generally useful modifications as well as suggestions, bug reports and requests for wanted features should be sent to the author. You may give away copies of the unmodified distribution archives, but not offer them for public access.

Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. The program is provided *as is* without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event will the author be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program or its documentation (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs or damages to your computer hardware), even if the author or another party has been advised of the possibility of such damages.

The program uses the ncurses (terminal versions on Unix platforms) or PDCurses (Dos platforms and X11 versions on Unix platforms) library, including the respective panel libraries. The DJGPP version of the program uses DJGPP code like the DJGPP libc.a and the protected mode stub and is distributed together with DJGPP code like the DPMI server '**cwsdpmi**' and the '**redir**' utility. The DJGPP code and its sources are available on SimTel/DJGPP mirrors around the world, e.g., from 'ftp://ftp.uni-paderborn.de/pub/simtelnet/gnu/djgpp/'.

The program uses the following libraries which are released under the GNU Library General Public License (see the file 'COPYING.LIB' – note that this license does **not** apply to the program):

– The GNU Pascal runtime library. Its source is available as part of the GNU Pascal compiler (GPC) distributions (which you will need to compile it, anyway) from 'http://home.pages.de/~GNU-Pascal/'.
– The rx library.

13 Reporting Bugs And Contact

There is a mailing list devoted to PENG, ⟨`peng-list@g-n-u.de`⟩. Mails to the list should be in English. To subscribe, send the command '`subscribe peng-list <your@mail.address>`' in the body of a mail to ⟨`peng-list-request@g-n-u.de`⟩. The list is archived on the WWW at '`http://www.gnu.de/archive/wilma.cgi/peng-list`'.

If you think you've found a bug, anything from a typo to a segfault (or system crash under Dos ;–), please report it on the list. If necessary, you can also contact the author directly at ⟨`frank@g-n-u.de`⟩. Please include an exact description of the bug and the circumstances under which it shows as well as the version (release and platform) of PENG. Since PENG is a complex program, it is important to describe the circumstances exactly, otherwise I might not be able to reproduce and fix the bug. Sometimes it might be necessary to include one or several (data or config) files to reproduce the bug. In this case, please keep the files small by removing parts irrelevant to the bug, and uuencode or MIME encode the files if necessary. Suggestions for new features can, of course, also be addressed to the list or the author.

'HOST'
'LOGNAME' The name of the host and the user, used for creating a 'vim' compatible header for the lock files (see Section 10.7.2.4 [Menu Options/Editor2/Use lock files], page 46).

'EDITOR' Not used by PENG, but if you set it to 'peng', many programs will invoke PENG as their default editor. Highly recommended. :-)

The following variables are specific to PENG:

'EDNAME'
'EDNODE'
'COL'
'LINE'
'CURTOKEN'
'CURLINE'
'ERRNAME'
'ERRCOL'
'ERRLINE'
'TABSIZE'
'DIR_SEPARATOR'
Automatically set by PENG before invoking a tool (see Section 11.3 [Automatic Variables], page 65).

any other variables
which are defined under Options/Variables (see Section 10.7.6 [Menu Options/Variables], page 51) or used by some tools (see Chapter 11 [Tools], page 63).

12 Environment Variables Used By PENG

PENG uses the following environment variables. The variables listed in the first table are standard variables used by many or most programs (under Unix, anyway), so if you have a properly installed system, you shouldn't have to worry about them.

‘SHELL’
‘COMSPEC’ (Dos)
: The name of the OS shell invoked by ‘File/OS shell’ (see Section 10.1.13 [Menu File/OS shell], page 26) and used to execute PENG tools (see Chapter 11 [Tools], page 63).

Under Dos, PENG will try ‘SHELL’ first, using ‘-c’ to specify a command (which is what ‘bash’ understands, so if you installed the DJGPP version of ‘bash’, and set ‘SHELL’ to its path, PENG will use ‘bash’). If ‘SHELL’ is not set, PENG will use ‘COMSPEC’ (which is usually set to the path of ‘command.com’), using ‘/c’ to specify a command (which is what ‘command.com’ understands).

‘PATH’
: A list of directories where various executables are searched, e.g., the printer spooler (Unix), or the uncompressor for a compressed info file (see Section 10.9.6 [Info Node Format], page 60).

‘TEMP’
‘TMP’
‘TEMPDIR’
‘TMPDIR’
: Tried (in this order) when PENG needs to create a temporary file name, e.g., for the ‘TEMP’ tool function (see Section 11.6.21 [Tool TEMP], page 71), or when uncompressing compressed info files (see Section 10.9.6 [Info Node Format], page 60).

‘HOME’
: The name of your home directory. Used to find the per-user default config file, to expand ‘~’ and to create temporary files if no appropriate variable (see above) is set and no writable system-wide temp directory (under some standard names) could be found.

‘INFOPATH’
‘INFODIR’ (if ‘INFOPATH’ is not set)
: The path to the installed info documentation files of PENG and other programs.

‘TERM’ (Unix, terminal)
: The type of the terminal you are working on. Setting this incorrectly will often lead to a corrupted display, or (some) function keys or key combinations not working correctly or, e.g., the number of lines on the screen not being recognized correctly (see Section 5.2 [Terminal Trouble Shooting], page 9).

‘TERMINFO’ (Unix, terminal)
: A path to the ‘terminfo’ terminal description files installed on your system. Setting this incorrectly will lead to a message like ‘Error opening terminal: linux.’ when starting PENG (see Section 5.2 [Terminal Trouble Shooting], page 9).

‘ESCDELAY’ (Unix, terminal)
: The number of milliseconds allowed between an ‘Esc’ character and the rest of an escape sequence. Setting it to a value too small can cause problems with PENG not recognizing escape sequences such as function keys. Setting it to a value too large can delay the recognition of an ESC press notably (see Section 5.2 [Terminal Trouble Shooting], page 9).

‘DISPLAY’ (Unix, X11)
: The name of the X11 server to connect to.

'M' (S,I,T) ("Message")
: Like 'm', but automatically seek to the position of the first message in an editor window if 'Options/Message window/Automatically go to first message' (see Section 10.7.8.4 [Menu Options/Message window/Automatically go to first message], page 51) is enabled.

'p', 'P' (I,T) ("parallel")
: Like 'm' and 'M', respectively, but PENG will immediately continue tool execution and interactive mode while the process is running (only on multitasking systems, of course). Only possible when redirecting both output and error this way, and only with 'SHELL', not 'STATUS', because the process keeps running, so its output, error and status can't be retrieved during the tool's execution. 'B' will seek to the position of the first message when the message window is in or comes to the foreground after the process has finished.

'b', 'B' (I,T) ("background")
: Like 'p' and 'P', respectively, but put the message window in the background. It can, of course, be brought to the foreground and back again at any time while the process is running (on multitasking systems).

11.8 The Built-In PENG Tools

PENG has some built-in tools for compiling Pascal and C programs with 'GPC' and 'GCC' and writing 'LaTeX' and 'Texinfo' texts, and some macros for those languages, and various other useful things.

They are used by default, but you can disable them if you don't want them (see Chapter 8 [Command Line], page 21). You can also dump them to a file (see Chapter 8 [Command Line], page 21), and use this as a base for your own configuration. You are explicitly allowed to modify these tools. However, note that it might make upgrading to a new PENG version a little harder because the new version will probably come with more or less different built-in tools and you might have to adapt your modifications accordingly. If you send useful modifications to me (and give me permission), I will include them in future PENG distributions.

The tools are available in the 'Run' (see Section 10.4 [Menu Run], page 32), 'Compile' (see Section 10.5 [Menu Compile], page 32), 'Tools' (see Section 10.6 [Menu Tools], page 34), 'Options' (see Section 10.7 [Menu Options], page 43) and 'Help' (see Section 10.9 [Menu Help], page 59) menus and are documented in this manual. They are differentiated from the unchangeable menu items by a header line 'The PENG Built-In Tool...', and the corresponding variables by a header line 'The PENG Built-In Tool Variable...'

Please note: Many of the tools expect certain programs to be installed on the system – sometimes particular programs for certain purposes (e.g., compilers for the different languages), and sometimes generic shell, text, file or other utilities. Therefore, if you are using the tools on a system without appropriate tools (e.g., Dos without a complete DJGPP installation), don't expect everything to work.

'S' (S) and 's' ("screen")
: The output is displayed full-screen and the executed process is given control of the terminal. On non-multitasking systems, this is the only way to run interactive programs and see their output while they're running.

'K' (S) and 'k' ("key")
: Like 's', but after running the program, the user must press ⟨ENTER⟩. This is the default for the main command. In the X11 version, pressing ⟨ENTER⟩ is not required, since PENG runs in its own window, anyway.

'O' (S) and 'o' ("ok expected")
: Like 'k', but the ⟨ENTER⟩ press is only required if the invoked program fails, i.e., returns non-zero status.

'E' (S) and 'e' ("error expected")
: Like 'k', but the ⟨ENTER⟩ press is only required if the invoked program succeeds, i.e., returns zero status.

'w' (S,I,T) ("window")
: The output is redirected to a newly opened (read-only) editor window, unless empty.

'i' (S,I,T) ("insert")
: The output is inserted into the current editor window at the cursor position.

'm' (S,I,T) ("message")
: The output is read into the message window (see Section 10.8.2 [Menu Window/Messages], page 58). Not possible in combination with a screen redirection ('s', 'S', 'k', 'K', 'o', 'O', 'e', 'E').

  The output must be in GNU standard message format (see section "Errors" in *the GNU Coding Standards*), with three extensions:

  - PENG recognizes multiple lines of messages as belonging together if their file names, line and column (if any) numbers match, and all lines but the first contain extra spaces before the message text.
  - PENG recognizes *progress messages* by programs to indicate their activity while running. Progress messages must be on lines by their own, and be prefixed with '#progress#'. New messages override the previous ones by default. However, a line consisting of exactly '#progress+#' indicates the beginning of a new level of messages (which will be shown below the previous messages). A line consisting of '#progress-#' indicates the ending of a level and causes the messages of that level to be removed.
  - PENG can display a *progress bar* displaying a percentage of the job done. Messages for the progress bar must be on lines by their own, and consist of '#progress-bar#', followed by a number between 0 and 1. A line consisting of '#progress-bar+#', followed by 3 numbers, *Offset*, *Size* and *Range*, will cause further values to be scaled, according to the formula

    Value = *Offset* + *Size* * *Given_Value* / *Range*,

    where the given values are expected to be between 0 and *Range*. Multiple levels of nested scaling are allowed. One level of scaling is terminated with '#progress-bar-#'. Scaling is useful to fit the progress bar values of sub-processes into a global process bar. All numbers can be in either integer or real format.

  If you want to use programs that write their messages in another format, you can pipe them through a converter. An example of such a converter is the included 'latexmsg' script (see Chapter 3 [Files], page 5) for the messages of 'LaTeX' whose format differs quite widely from GNU standard message format.

11.6.42 'CHDIR' (1)

Changes to the given directory. Does not return anything.

11.6.43 'CONFIRM' (*)

Displays the given text to the user and waits for a 'y'/'n' answer. If the answer is 'n', or the question is aborted with ⟨ESC⟩, the tool operation is aborted, otherwise is just continues.

Does not return anything.

11.6.44 'ERROR' (*)

Shows the argument as an error message to the user and aborts the tool operation.

Does not return anything.

11.6.45 'WARNING' (*)

Shows the argument as a warning to the user, but does not abort the tool operation.

Does not return anything.

11.7 Redirecting The Output Of Shell Tools

Output redirections are possible for the arguments to the 'SHELL' (see Section 11.6.30 [Tool SHELL], page 74) and 'STATUS' (see Section 11.6.31 [Tool STATUS], page 74) functions and for the *main command* (the whole tool action). They are of the form '#1', '#2' or '#' (and therefore can be escaped by single or double quotes) to redirect the output, error, or both of them, respectively, and must be followed by the destination (see below).

Please note: Normal redirections with '>' are left to the shell to be handled in the usual way. Under Dos, bash style redirections are automatically converted into 'redir' calls. Redirections of the shell and of PENG are each evaluated from left to right. However, redirections of the shell always take precedence before those of PENG, so 'foo > bar #i' will redirect the output of 'foo' to 'bar' and not insert it into the current editor window.

The following destinations are recognized. An 'S' in parentheses means, if the process returns non-zero status, the tool operation is aborted (not with 'STATUS'). An 'I' means, the process gets an empty input. A 'T' means, PENG keeps control of the terminal, and the process (together with its child processes) can be aborted by ⟨ESC⟩ (of course, this does not apply under non-multitasking systems). All of these properties apply for unredirected output/error. They are effective only when they apply to both the actual output and the actual error redirection.

11.6.41 'INSERT' (*)

Inserts the given text into the current editor window at the cursor position, and does not return anything.

C-like escape sequences (esp. '\n', also '\007') are recognized as well as the following special sequences:

'\\' This yields a single '\'.

'\i' This sequence is replaced by the indentation of the current line.

'\I' This sequence is replaced by the indentation given under 'Options/Editor2/Indent width' (see Section 10.7.2.16 [Menu Options/Editor2/Indent width], page 48).

'\N' Equivalent to '\n\i'.

'\C' This sequence is not inserted, but causes the cursor to be placed at this position after insertion.

'\c' This sequence is not inserted, but the position is remembered for jumping to it via ⟨CTRL⟩-⟨UP⟩, ⟨CTRL⟩-⟨DOWN⟩ (see Section 5.9 [Text Editor Keys], page 13).

These sequences make it possible to define tools that insert, e.g., an indented block into a program and place the cursor at a certain position within the block and provide an easy way to jump to other important positions (see Section 11.6.41.1 [Syntax Macros], page 76).

11.6.41.1 PENG Tools Used As Syntax Macros

The 'INSERT' tool function (see Section 11.6.41 [Tool INSERT], page 76) makes it possible to define PENG tools that provide syntax macros for various languages.

The PENG built-in tools provide several macros for Pascal, C, LaTeX and Texinfo. They can be found in the 'Tools/Macros/*LANGUAGE*' menus (see Section 10.6.1 [Menu Tools/Macros], page 34).

All of the syntax macros are only active for files whose extension matches some common extensions used for the particular language.

Therefore, the macros for different languages can share the same hotkeys. All the macro hotkeys use the ⟨EXTRA⟩ key, so if you have a system with a working ⟨EXTRA⟩ key (see Section 5.1 [Modifiers], page 9), these macros can save you a lot of typing.

Whenever possible, the hotkeys are chosen so that similar constructs use the same hotkey in all languages, so people who often program or write in different languages don't have to remember too many different hotkyes. (Ideally, they could write their program just by using hotkeys without caring about the language at all, but we're not quite there yet... ;-). All languages have a 'New program' (or 'New text') macro which is best used immediately after opening a new empty file, with an ⟨EXTRA⟩-⟨N⟩ hotkey, and e.g., the 'case' statement in Pascal and the 'switch' statemtent in C share the same hotkey ⟨EXTRA⟩-⟨C⟩.

1 Introduction

PENG is a portable Visual Text Editor, Integrated Development Environment, Textual User Interface, or whatever TLA you prefer.

PENG was written completely in Pascal and created using the GNU Pascal compiler, and is therefore portable to many systems, including most Unix compatible systems and the DJGPP Dos extender.

Like many program names, the acronym PENG is compiled from some descriptive words. However, unlike most other names, it is not statically linked, but instead you, the user, can link it dynamically from the following compilation of words. However, please note that the name 'PENG' is not licensed freely, so you may not change or add words without the author's permission.

P: PENG, Pascal, Portable, Powerful
E: Editor, Equipment, Environment, Enjoyment
for
N: New, Noble, Nonconformist, Natural
G: Generations, Gnus, Gurus, Galaxies

The Unix versions of PENG do not require anything special to be installed, just the usual utilities and shared libraries (libc, libm, and libncurses including libpanel and terminfo for the terminal version, or some X11 libraries for the X11 version) and, of course, an X11 server for the X11 version. All of these things should be present on a properly installed Unix (compatible) system (except maybe X11, but in this case, you probably don't want to use the X11 version, anyway ;-).

The DJGPP version is functionally mostly identical to the Unix versions, with a few limitations imposed by Dos, like lack of multitasking, so you can't start PENG tools in the background. You do not *have* to install the DJGPP Dos extender itself in order to use the DJGPP version of PENG. Details are described in a separate chapter.

[Tool SHELL], page 74) execution, they're automatically reloaded without asking for confirmation (unless the editor window contains unsaved modifications).

Does not return anything.

11.6.36 'VIEW' (any)

Like 'OPEN' (see Section 11.6.35 [Tool OPEN], page 74), but opens the editor windows read-only if they are not already open.

Does not return anything.

11.6.37 'INFO' (any)

Opens info reader(s) with the given node names (see Section 10.9.6 [Info Node Format], page 60) if they exist.

Does not return anything.

11.6.38 'CLOSE' (any)

Closes all the files that are opened in an editor window and whose name (with or without directory) matches one of the patterns given. If no arguments are given, the current editor window, if any, is closed. If a window's contents are modified, the user is asked whether to save them. If this question is aborted, the whole tool operation is aborted.

Does not return anything.

11.6.39 'SAVE' (any)

Saves all the files that are opened in an editor window and were modified and whose name (with or without directory) matches one of the patterns given (so '`$(SAVE *)`' will save all files opened). If no arguments are given, the current editor window, if any, is saved if it was modified.

Does not return anything.

11.6.40 'GOTO' (1 or 2)

Goes to the given line [and column] in the current editor window.

Does not return anything.

11.6.30 'SHELL' (*)

Invokes a system command with arguments by calling your shell (the one that is given in the SHELL environment variable). Unlike the other functions, quotes are honoured, but not removed here, so you don't have to double-quote special characters to prevent the shell from expanding them. Variable evaluations and function calls are evaluated before calling the shell, however, so you do have to quote '$' signs meant for the shell unless they occur within single quotes. The output and error of the command, unless redirected (see Section 11.7 [Output Redirections], page 77), is returned, with multiple lines separated by spaces (just like backquotes do in the usual shells).

11.6.31 'STATUS' (*)

Like 'SHELL' (see Section 11.6.30 [Tool SHELL], page 74), but returns the status rather than the output and error of the command. I.e., a successful command returns '0', an unsuccessful one returns a number not equal to 0, a command terminated/stopped by a signal returns 'Signal *42*' or 'Stopped *42*', respectively, where *42* is the number of the signal. The output and error of the command are discarded, unless redirected (see Section 11.7 [Output Redirections], page 77).

11.6.32 'EVAL' (*)

Evaluates the argument like the tool actions do (see Section 11.4 [Tool Evaluation], page 66). Can be used, e.g., to let variables refer to other variables by '$*VARNAME*' or even to some of the tool functions (see Section 11.6 [Tool Functions], page 67). The expression to be evaluated can contain 'EVAL' calls again, to a maximum depth of (currently) 16 – this is to prevent endless recursion as in '$(SET foo '$(EVAL $foo)') $(EVAL $foo)'.

11.6.33 'IGNORE' (*)

Ignores its argument and returns nothing. Useful if you don't care for the result of other tool functions such as 'SEARCH' (see Section 11.6.26 [Tool SEARCH], page 72)

11.6.34 'SET' (2)

Sets an environment variable with the name given in the first argument to the value given in the second argument. These variables are persistent while PENG runs. This can be useful, e.g., to have a *Compile* tool which stores the name of the generated executable in a variable, and a *Run* tool which runs the executable with the 'SHELL' (see Section 11.6.30 [Tool SHELL], page 74) function. If a PENG variable of that name is defined, and has 'Read from environment' enabled, its value is also updated.

Does not return anything.

11.6.35 'OPEN' (any)

Opens the given file names as editor windows. If they are already open, they are just brought to the top and if the files on disk were modified, e.g., by a previous 'SHELL' (see Section 11.6.30

2 Some Highlights

- Portability. In particular, there are versions for general terminals on Unix systems, for X11, and for DJGPP. The terminal versions work naturally across telnet/rlogin/slogin between systems with the same or different terminal types, and X11 is network-transparent, anyway.
- Support for multiple windows, including editing of the same file in several windows at the same time
- Advanced block operations, persistent and non-persistent blocks, editable clipboard, automatic copy function
- A reader for GNU Info documentation, including support for compressed info files
- A paragraph mode, useful for editing mail, news, TeX, HTML, etc.
- A quote protection mode for editing mail and news
- Multi-level Undo/Redo capability
- Lex highlighting
- Autosave mode
- Combines the ease of use known from some Dos IDEs with the power of Unix environments, e.g.:
  - Supports function keys (where available on the terminal) and file selection boxes, but also file name completion with Tab; '$' for environment variables as well as '~' and '~user' for home directories, and all the file name wildcards like '*', '?', '[^-]', '{,}'
  - Easy search/replace capabilities, but also regular expression searches and replacements with backreferences
  - Handles files with Unix or Dos line endings on both kinds of systems, without converting them automatically, but with the easy possibility to convert them explicitly at any time.
  - You can invoke an OS shell, but you can also suspend PENG.
  - You can use control characters like Control-C for cursor movement in PENG if you want, but PENG still reacts correctly to external interrupt, termination or hangup signals and saves unsaved changes whenever possible.
  - Readable configuration files in text format, a global one and per-user ones under Unix
  - Simple and numbered backup files
  - Lock files (compatible with vim) to prevent several editors from editing the same file at the same time
  - Easily accessible context-sensitive help, but in the same format (GNU Info) as used by many other programs, and easily extendable for your own tools
  - PENG tools can invoke external processes (e.g., compilers), but they can also fork them to run in the background while you can already go on editing your files.
- Completely configurable colors. Very important. :–)
- A way to open files read-only, from the menu and the command line
- No mouse support. (–: Leaves the mouse to gpm and similar applications.)
- Hotlist for easy access to files edited frequently
- Extensive tool scripting capabilites:
  - Powerful syntax, similar to Unix shells
  - User-defined variables of several types
  - Many pre-defined functions for strings, regular expressions, file names, boolean values, mathematical operations on real and complex numbers, interaction with the user and control of the editor and info reader

- Conditional statements
- Access to a message window, suitable for GNU standard message format, but other message formats can be converted. An example converter for '`LaTeX`' messages is included.
- PENG can start external processes in the background and still receive their output. Background processes can be killed easily. Very useful for long compilations.
- Tools can depend on the file name of the current editor and on other conditions, to allow e.g. several versions of a similar tool by the same name for different languages/formats.
- PENG can display progress messages and a progress bar while running tools if the tools support it.
- Tools can be executed automatically at the start or termination of PENG.
- Many built-in tools, including usage of utilities like '`grep`', syntax macros, and the compiler invocations with hotkeys familiar from other IDEs (for '`GPC`' and '`GCC`' and very similarly even for '`LaTeX`' and '`makeinfo`'), all fully customizable

11.6.27 '`MATH`' (*)

Evaluates a mathematical expression in real or complex numbers. The expression may consist of

- real numbers, using the *e* notation,
- the constants '`e`', '`pi`', '`i`',
- the operators '`+`', '`-`', '`*`', '`/`', '`^`',
- the functions '`abs`', '`sqrt`', '`sin`', '`cos`', '`tan`', '`arctan`', '`exp`', '`ln`',
- parentheses ('`()`').

The expression is evaluated according to the usual mathematical rules (which are not quite the same as those in most programming languages). E.g., a '`*`' may be omitted between two expressions, the second of which is not a numeric constant, and arguments to functions do not need parentheses unless they are necessary to override precedence rules.

Precedence, from highest to lowest:

- parentheses
- numeric and symbolic constants, unary functions
- '`^`',
- '`*`', '`/`'
- '`+`', '`-`'

Please note: Due to the *e* notation, there is an ambiguity with terms like '`2e`' which will be interpreted as '`2*10^`...'. If you mean '`2*e`', write so, or '`2 e`' (with a space between the '`2`' and the '`e`'). Expressions like '`3e+4+5`' can be confusing, but are interpreted according to the *e* notation (i.e., this expression equals 30005).

11.6.28 '`RANDOM`' (0 .. 2)

If called with no arguments, '`RANDOM`' returns a random real number between 0 (inclusively) and 1 (exclusively). If called with 1 argument, which must be a positive integer *n*, '`RANDOM`' returns a random integer number between 1 and *n* (inclusively). If called with 2 arguments, which must be integers *m* and *n* with *m* `<=` *n*, '`RANDOM`' returns a random integer number between *m* and *n* (inclusively).

11.6.29 '`TRYINFO`' (1)

Tries to open an info reader with the given node name (see Section 10.9.6 [Info Node Format], page 60), like '`INFO`' (see Section 11.6.37 [Tool INFO], page 75). Returns '`True`' if successful, and '`False`' otherwise.

11.6.23 'YNQUESTION' (*)

Displays the given text to the user and waits for a 'y'/'n' answer. If the answer is 'y', it returns 'True'. If the answer is 'n', it returns 'False'. If the question is aborted with ⟨ESC⟩, the tool operation is aborted.

11.6.24 'SELECT' (2 or more)

Displays a menu with the first argument as the title, and the further arguments as the items, and lets the user choose one item. Returns the item chosen. If the menu input is aborted, the whole tool operation is aborted.

11.6.25 'SUBST' (2 .. 4)

Takes the first argument and replaces all matches of the second argument, treated as a basic regular expression (see Section 10.3.6 [Menu Search/Regular expression], page 29), by the third one with backreferences and C-like escape sequences (esp. '\n', but not '\007') allowed (see Section 10.3.2 [Menu Search/Replace], page 28), or by nothing if only two arguments are given. '\\' yields a single '\'. The optional fourth argument can contain a combination of the following flags:

'e' The second argument is treated as an extended regex.

's' The second argument is treated as a plain string, rather than a regex.

'i' The matching is case-insensitive.

'w' Only whole word matchings are found.

'1' Only the first match, if any, is replaced.

11.6.26 'SEARCH' (1 or 2)

Searches the first argument, treated as a basic regular expression (see Section 10.3.6 [Menu Search/Regular expression], page 29), in the current editor window. If it is found, it goes to the position in the editor and returns 'True', otherwise it returns 'False'.

The optional second argument can contain a combination of the following flags:

'e' The first argument is treated as an extended regex.

's' The first argument is treated as a plain string, rather than a regex.

'i' The matching is case-insensitive.

'w' Only whole word matchings are found.

'f' Search only from the current cursor position.

'a' Search only after the current cursor position.

'b' Search backwards.

'm' Search marked text only.

3 Files In The Binary Distribution

The following list describes the files distributed in the PENG binary archives. It also contains the installation instructions by simply describing what to do with each file.

'README' A text file with the most important information about PENG. Read it before starting PENG. Most of the information is also contained in the PENG info documentation.

'COPYING.LIB'
The license applies to some of the libraries used in PENG. This license **does not** apply to PENG itself! See the chapter about legal stuff.

'peng' (Unix, terminal)
'xpeng' (Unix, X11)
'peng.exe' (DJGPP)
The program executable. You can install it anywhere you want (provided you or your 'PATH' will find it again ;–).

'peng.info'
The PENG online documentation. Copy it into one of your info directories, i.e., '/usr/info' or '/usr/local/info' or a directory named in your 'INFOPATH' (or 'INFODIR') environment variable.

'latexmsg'
A shell script to convert 'LaTeX' messages to GNU message format. This is used by the built-in 'Compile/Make' and 'Compile/Run' tools for 'LaTeX' files in order to invoke 'LaTeX' from PENG just like 'GPC' and 'GCC'. If you want to use it, just copy it to somewhere in your 'PATH'. Please note that this script is distributed under a different license than PENG. This file is not part of PENG, and may be modified and distributed separately.

'terminfo-linux.tar.gz' (Unix terminal versions only)
This is a patch to enable PENG and other ncurses programs to make use of the ability of Linux 2.2 and newer kernels to produce a block cursor when needed (e.g. for PENG's overwrite mode). Though PENG can simulate a block cursor with graphic characters (see 'Options/Environment/Simulate block cursor'), a real block cursor looks better, of course. The present patch can be installed without recompiling anything, just by copying some files into place. More details can be found in the 'README' file included in this archive. The patch will not do any harm on older kernels. Please note that **not** only on Linux machines it is useful to install the patch. Installing them on any other machine will allow users who telnet in from a Linux console to profit from the block cursor capability. Besides, some Unix systems have installed older Linux terminfo entries or none at all, so it's a good thing, anyway, to give them a current version. The patch is included in the terminfo database of ncurses 5.0. However, since ncurses 5.0 was released quite recently (1999-10-23), it can't be expected to be present on many systems in the near future, so I include the patch with PENG. The new terminfo files do not require ncurses 5.0, so you can install them on a system with older ncurses if you don't want to upgrade ncurses. This file is not part of PENG, and may be modified and distributed separately.

11.6.21 'TEMP' (0)

Returns a unique temporary filename. It is guaranteed to not exist and to be distinct each time the function is called. The function does not create a file or directory, it just returns the name.

11.6.22 'PROMPT' (1 .. 3)

Displays a prompt to the user, with the first argument as its title, and the second argument, if given, as the initial content – otherwise, the word under the cursor in the current editor window, if any, is used as the initial content, unless the 'w' flag (see below) is given. Returns what the user enters. If the user aborts the prompt with ⟨ESC⟩, the whole tool operation is aborted. The optional third argument can contain a combination of the following flags:

'w' Don't copy the initial content from the current editor window and don't allow the user to copy characters from the current editor window.

'u' Convert the result to upper case.

's ("small")'
Convert the result to lower case.

'l' Trim leading spaces.

't' Trim trailing spaces.

'e' Expand the result to a full path name.

'v' Remove the directory if it is the current one.

'r' Remove a trailing directory separator from the result.

'f' Use a file selection box if the string contains wildcards.

'd' Use a directory selection box if the string contains wildcards.

'c' Enable file name completion in the prompt. This is automatically implied by 'f' and 'd'.

'm[:*x*]' Handle multiple file names, separated by *x* (a single character) if given (default: ' '). This affects 'l', 't', 'e', 'v', 'r' and file name completion.

'p[:*x*] ("path")'
Handle multiple file names, separated by *x* if given (default: ':').

'a' With 'f' or 'd', don't start the selection box with the default mask if an empty string is entered.

'b' With 'f' or 'd', don't prompt, but immediately start the selection box.

'=foo' With 'f' or 'd', specify the default mask (default: '*') in the selection box when an empty string is entered and 'a' is not given, or a directory name is entered for 'f'. This option must be the last one, because the rest of the option string, following the '=', is regarded as the default mask.

At most one of 'f', 'd', 'm' or 'p' is allowed.

11.6.13 'MATCH' (1 or more)

Returns 'True' if the file name given in the first argument (including directory if present!) matches any of the masks given in the remaining arguments, and 'False' otherwise.

11.6.14 'GLOB' (any)

Returns all file names that match one of the patterns given.

11.6.15 'DIR' (any)

Returns the directory of the argument(s). The result will always end with a directory separator – if a file name contains no directory, the symbol that stands for the current directory ('.' on most systems), followed by a directory separator, is returned.

11.6.16 'NAME' (any)

Returns the base name without extension of the argument(s).

11.6.17 'EXT' (any)

Returns the file name extension of the argument(s).

11.6.18 'BASENAME' (1 or 2)

If called with one argument, it returns the base name with extension. If called with two arguments, and the second argument is a suffix of the result, it also removes this suffix, just like 'basename(1)' does (see section "'basename' Strip directory and suffix from a file name" in *the GNU shell utilities manual*).

11.6.19 'REALPATH' (any)

Returns the argument(s) with all relative paths and occurrences of the symbols that stand for the current and parent directories ('.' and '..' on most systems) and symbolic links resolved in the directory, i.e., the path without the file name. Under Dos, it converts the result to lower case and removes a trailing '.', like Dos does. Arguments whose directory is invalid are ignored.

11.6.20 'CHANGEEXT' (2)

Removes the extension from the first argument and appends the second one.

4 Special Notes About The DJGPP (Dos) Version

This chapter describes some things to note when running PENG under Dos. If you want to run PENG under Dos, *please* read it. If you're running PENG on other systems only, you can safely skip this chapter.

First of all, if you want to read the PENG online documentation (and I guess you do :–), you might have to set up the 'INFOPATH' environment variable first:

If you have DJGPP installed, copy the file 'peng.info' (see the list of files in the binary distribution) into '%DJDIR%\info', and add some lines like the following to your '%DJDIR%\djgpp.env':

```
--- cut ---
[peng]
INFOPATH=%/>;INFOPATH%%DJDIR%/info;%DJDIR%/gnu/emacs/info

--- cut ---
```

(The second line should be identical to the one in the '[info]' and/or '[emacs]' sections.)

If you *don't* have DJGPP installed, copy 'peng.info' (see the list of files in the binary distribution) to some arbitrary directory (say, 'c:\usr\info'), and point the environment variable 'INFOPATH' to that directory, e.g. with a statement like the following in your 'autoexec.bat':

```
set INFOPATH=c:\usr\info
```

On Dos systems without "long" file name support, you might have to rename 'peng.info' to 'peng.inf', which your archive extractor probably will have done automatically. (Oh yeah, 9 characters is so terribly long, ROTFL... ;–)

The DJGPP version is functionally mostly identical to the Unix versions, with a few limitations imposed by Dos, like lack of multitasking, so you can't start PENG tools in the background. To use the DJGPP version of PENG, you do not have to install the DJGPP Dos extender itself (though this is certainly recommended if you still have to use Dos). The files provided in this archive should be sufficient to run PENG under Dos. However, you might want other DJGPP tools for certain functionality, such as the 'gunzip' uncompressor to read compressed info files. DJGPP is available from SimTel/DJGPP mirrors around the world, e.g., from 'ftp://ftp.uni-paderborn.de/pub/simtelnet/gnu/djgpp/'. If you encounter any problems, you might want to read the DJGPP FAQ, available, e.g., at 'ftp://agnes.dida.physik.uni-essen.de/gnu-pascal/djgpp/faq211b.zip'.

The following files which are part of DJGPP, not of PENG, are distributed with the DJGPP versions of PENG for your convenience. If you have installed DJGPP, you've probably got those files already.

'cwsdpmi.exe'
: A DPMI server that PENG can use. PENG can also use the DPMI server that comes with recent Dos versions, but in my experience, 'cwsdpmi' often works better. If you want to use it, just copy it to somewhere in your 'PATH'.

'cwsdpmi.doc'
: The documentation of 'cwsdpmi'.

'redir.exe'
: A DJGPP utility for I/O redirection. PENG itself does not need it, but it will try to use it (transparently) when executing PENG tools that use bash style redirections like '&>' or '2>'. Therefore, it is recommended to install 'redir.exe'. To do so, just copy it to somewhere in your 'PATH'.

Some of the built-in PENG tools were written mainly for Unix systems. Most of them should work under DJGPP as well, provided the programs and utilities used by the tools (e.g. 'grep', 'sed', GNU textutils, and in particular 'bash', all available in the DJGPP distributions) are installed, but this has not been tested yet. The 'latexmsg' script, used by a few of the tools, might work on a DJGPP system with 'bash' and 'LaTeX' installed, but I cannot test this currently. If you try any of these things, please tell me about your results.

'l' The last match is found rather than the first one.

11.6.8 'EQ' (1 or 2)

Returns 'True' if the two arguments are equal as strings (or, if only one argument is given, if it's empty), and 'False' otherwise.

11.6.9 'NE' (1 or 2)

Returns 'True' if the two arguments are not equal as strings (or, if only one argument is given, if it's not empty), and 'False' otherwise.

11.6.10 'AND' (1 or more)

Returns 'True' if all the arguments are 'True', and 'False' otherwise. All the arguments must be either 'True' or 'False' (case-insensitively and ignoring any leading or trailing spaces).

Please note: 'AND' does **not** perform *shortcut evalutaion*, i.e., even if the first argument is 'False', the remaining arguments are still evaluated. If you need shortcut evaluation, you can achieve this using '#if' statements (see Section 11.5 [Conditional Statements], page 67). E.g., the following statement will have the effect of a shortcut 'AND' with two arguments (though, in some cases, there exist shorter alternatives, e.g., if the 'AND' is the whole condition of an '#if' statement without an '#else' branch):

```
#if cond1 #then cond2 #else False #fi
```

11.6.11 'OR' (1 or more)

Returns 'True' if at least one of the arguments is 'True', and 'False' otherwise. All the arguments must be either 'True' or 'False' (case-insensitively and ignoring any leading or trailing spaces).

Please note: 'OR' does **not** perform *shortcut evalutaion*, i.e., even if the first argument is 'True', the remaining arguments are still evaluated. If you need shortcut evaluation, you can achieve this using '#if' statements (see Section 11.5 [Conditional Statements], page 67). E.g., the following statement will have the effect of a shortcut 'OR' with two arguments (though, in some cases, there exist shorter alternatives, e.g., if the 'OR' is the whole condition of an '#if' statement without an '#else' branch):

```
#if cond1 #then True #else cond2 #fi
```

11.6.12 'NOT' (1)

Returns 'True' if the argument is 'True', and 'False' otherwise. The argument must be either 'True' or 'False' (case-insensitively and ignoring any leading or trailing spaces).

Procedures are called like functions and take arguments like them and perform some action, but do not return anything.

11.6.1 'UPCASE' (*)

Converts the argument to upper case.

11.6.2 'LOCASE' (*)

Converts the argument to lower case.

11.6.3 'TRIM' (*)

Removes leading and trailing spaces from the argument.

11.6.4 'LTRIM' (*)

Removes leading spaces from the argument.

11.6.5 'RTRIM' (*)

Removes trailing spaces from the argument.

11.6.6 'SUBSTR' (2 or 3)

Returns a substring of the first argument, starting from the position given in the second argument (which must be a number) with the length given in the third argument (also a number), or extending up to the end of the first argument if only two arguments are given.

11.6.7 'INDEX' (2 .. 5)

Finds the first match of the second argument, treated as a basic regular expression (see Section 10.3.6 [Menu Search/Regular expression], page 29), in the first argument, and returns the index of the beginning of the match. If no match is found, it returns 0. The optional third argument, if not empty, can contain an index where to start matching (default 1), and the optional fourth argument an index where to stop matching (the default is the length of the first argument). The optional fifth argument can contain a combination of the following flags:

'e' The second argument is treated as an extended regex.

's' The second argument is treated as a plain string, rather than a regex.

'i' The matching is case-insensitive.

'w' Only whole word matchings are found.

5 Keys And Terminals

PENG was mainly tested on PCs (Linux console and DJGPP), so there's no guarantee that other terminals will support enough function keys to be useful, but generally the key bindings should be portable.

5.1 Modifier Keys On Different Systems

ALT means the Alt key, but on some terminals it's the Meta key, or the ESC key which must be pressed before the other key. Under X11, it's modifier #1 (see 'xmodmap(1)').

ALTGR is the AltGr key on PC keyboards (Linux, DJGPP), and modifier #3 under X11.

EXTRA can mean another modifier key if present. On DJGPP, it's the Scroll Lock key, under X11 it's the modifier #4, and on a local Linux console, it's the 'CtrlL' modifier (value 64) which is unused on many keytabs and can be mapped e.g., to those keys on new keyboards with these ugly symbols waiting to be replaced by penguins (keycodes 125 and 127) by inserting the following two lines into your '/etc/default.keytab' and reloading the keytab with 'loadkeys':

```
keycode 125 = CtrlL
keycode 127 = CtrlL
```

5.2 Trouble Shooting For Terminal Problems (Unix)

The following points may help solve problems with the terminal versions on Unix systems.

- If it seems that the allowed time between the escape key and the following key is too small, or function keys or key combinations do not work reliably, especially on slow network connections, you can set the allowed time in milliseconds in the environment variable 'ESCDELAY' (in such cases, 1000 is often a useful value).
- On the other hand, if it takes a long time until an ESC key press takes effect, and you are on a fast connection or a local console, it might help to set 'ESCDELAY' to a low value.
- If the display is corrupted or (some) function keys or key combinations don't work at all or e.g., the number of lines on the screen is not recognized correctly, please check if your 'TERM' environment variable is set correctly and make sure an appropriate 'terminfo' entry is present on the host system.
- If you get an error message like 'Error opening terminal: linux.' when starting PENG, it might be that your terminfo database is in '/usr/lib/terminfo' (or perhaps '/usr/local/share/terminfo' or '/usr/local/lib/terminfo') rather than where PENG expects it ('/usr/share/terminfo' under, e.g., Linux, or '/usr/local/share/terminfo' under, e.g., Solaris). In this case, either create a symlink. E.g., under Linux, if your terminfo database is in '/usr/lib/terminfo', do (as root): 'ln -s /usr/lib/terminfo /usr/share/terminfo', or (as the user who wants to use PENG) set the environment variable 'TERMINFO' to the directory where your terminfo database lives. In the unlikely case that you don't have a terminfo database on your system at all, you'll have to install it as part of the ncurses library.

– On Linux consoles, if an editor window becomes corrupted when the opened file contains some non-printable characters (e.g., ASCII #128 and the following ones), you probably need to load a correct unimap with '`loadunimap(8)`'. This mostly affects speakers of languages with "high" characters (e.g., German) which load an appropriate font with '`setfont(8)`'. The '`loadunimap`' call should be done right after the '`setfont`' call in the system startup scripts, but at least one of the big German Linux distributions forgets the '`loadunimap`' call in its default configuration.

5.3 Alternative Keys

The following items list some key alternatives useful on terminals with limited function and special keys available. There are also some comments about keys on particular systems.

5.3.1 Alternative For The Shift Key

The SHIFT key combined with cursor movement keys, used for marking text, only works on local Linux consoles, DJGPP, and X11. As an alternative, CTRL-B can be used to virtually press/release SHIFT. However, this does not affect capitalization of letters.

5.3.2 Alternatives For Some Cursor Movement Keys

The following alternatives are provided for Wordstar/Borland compatibility and for terminals without function keys:

CTRL-E	UP
CTRL-X	DOWN
CTRL-S	LEFT
CTRL-D	RIGHT
CTRL-A	CTRL-LEFT
CTRL-F	CTRL-RIGHT
CTRL-R	PGUP
CTRL-C	PGDN
CTRL-G	DEL
CTRL-P	INS – this is not Wordstar/Borland compatible; they use CTRL-V, but CTRL-V is used for a different purpose in the text editor (see Section 5.9 [Text Editor Keys], page 13) which is compatible to '`bash(1)`' (see section "Commands For Changing Text" in *the bash manual*), '`csh(1)`' and '`vi(1)`'.

5.3.3 Alternatives For ALT Plus Function Keys

Some hotkeys use ALT-*Fn* key combinations. The Linux console uses these key combinations by default to switch virtual consoles, and some window managers (when running under X11) are also configured to use them for their own purposes. Therefore, PENG provides the alternatives (on all systems, of course) ALT-*n* for ALT-*Fn* (1 <= n <= 9) and ALT-0 for ALT-F10.

quotes with nothing between them. The values of variables and results of functions, when used as arguments to other functions, are separated into multiple arguments, unless the variable evaluation or function call itself is inside double quotes, but '`$`' signs and quotations within them are never touched. This is mostly like the usual shells behave. '`$`' can quote itself ('`$$`', like in makefiles) as well as single and double quotes. The latter is needed if you want to pass quote characters to the '`SHELL`' (see Section 11.6.30 [Tool SHELL], page 74) and '`STATUS`' (see Section 11.6.31 [Tool STATUS], page 74) functions without causing them to quote text from the PENG tool evaluation.

Characters that need quoting (in some or all situations) are currently: '`"`', '`'`', '` `', '`(`', '`)`', '`#`', '`$`'. Pairs of unquoted parentheses are allowed in an argument.

The resulting string, after all replacements, if not empty, is executed, i.e., treated like the arguments to the '`SHELL`' function (see Section 11.6.30 [Tool SHELL], page 74). This does not apply to tool conditions, where the resulting string is immediately taken as the value of the condition.

Please note: Tools like '`echo foo $(WARNING bar)`' will **first** give the warning '`bar`', and **then** write '`foo`' on the screen, because '`$(WARNING bar)`' is an argument to the main command. If you want the warning to be shown after the command is executed, you can do '`$(SHELL echo foo #K) $(WARNING bar)`' (See Section 11.7 [Output Redirections], page 77, about the meaning of '`#K`').

11.5 Conditional Statements in Tools

PENG tools understand conditional statements of the form

'`#if` *cond* `#then` *expr* `[#elif` *cond* `#then` *expr* ... `[#else` *expr*`]] #fi`'

The '`#elif`' and '`#else`' parts are optional. There can be any number of '`#elif`' parts and one '`#else`' part for each '`#if`'. Conditional statements can be nested to a maximum depth of (currently) 16. The conditions *cond* must evaluate to either '`False`' or '`True`' (case-insensitively and ignoring any leading or trailing spaces).

The whole conditional statement is replaced by exactly one of the *expr*s, depending on the conditions, or by nothing if no '`#else`' part is given and all conditions are false. In particular, '`#if`' can be used both for conditional statements (the usual '`if`' in programming languages) and for conditional values (the '`?:`' in C). Spaces immediately following the '`#if`', '`#then`', '`#elif`', '`#else`' and '`#fi`' directives are removed, however spaces before those directives are not.

Only the *expr* which is actually used is evaluated, i.e., functions in the other *expr*s are not called. This is, of course, what you would expect of an *if* statement.

11.6 Tool Functions

The following functions and procedures can be used in PENG tools.

The number of arguments is given in parentheses. '`*`' means the arguments are not separated, but treated as a single argument, including all the spaces except the first space after the function name.

'EDNODE' (1)
: The info node of the current info reader. Empty if the current window is not an info reader.

'COL' (1)
: The cursor column in the current editor window

'LINE' (1)
: The cursor line in the current editor window

'CURTOKEN' (1)
: The token (word) under the cursor in the current editor window

'CURLINE' (1)
: The content of the current line in the current editor window

'ERRNAME' (2)
: The file name of the last message selected

'ERRCOL' (2)
: The column of the last message selected

'ERRLINE' (2)
: The file line of the last message selected

'TABSIZE'
: The current tabulator size, given under '`Options/Editor2/Tab size`' (see Section 10.7.2.15 [Menu Options/Editor2/Tab size], page 48).

'DIR_SEPARATOR'
: The directory separator of the OS ('/' under Unix, '\' under Dos).

(1) empty if no window open or current window is not an editor

(2) empty if no current message

11.4 How Tools Are Evaluated

After setting the automatic variables (see Section 11.3 [Automatic Variables], page 65) and the defined environment variables (see Section 11.1 [Defining Variables], page 63), PENG evaluates the command line string from left to right, evaluating variable references and function and procedure calls.

Variable references are of the form '`$ENVVAR`' are replaced by the value of the environment variable. Both the automatic and user-defined variables and environment variables set before starting PENG can be used this way. If the variable is not set (as opposed to just being empty), this causes an error.

The variable name can also be enclosed in '{', '}'. In this case, special characters in the variable name don't have to be quoted with '\', and the name can be followed by a '-' or ':' and a value (e.g., '`${foo-bar}`'). That value is substituted if the variable is not set ('-'), or if it is not set or empty (':'). Empty variables are even supported under Dos, which can't handle empty environment variables.

Function calls are of the form '`$(FUNCTION arguments)`'. Arguments are separated by one of several spaces. See Section 11.6 [Tool Functions], page 67, for a list of available functions.

Parts of strings can be quoted with pairs of single or double quotes. Double quotes ('"') quote all characters except themselves and '$' (but including single quotes), single quotes (''') quote all characters except themselves. Empty arguments can be formed by pairs of single or double

5.3.4 Linux Keyboard Table

Besides ⟨ALT⟩-⟨Fn⟩ (see Section 5.3.3 [Alternative Alt], page 10), Linux consoles also use ⟨SHIFT⟩-⟨PGUP⟩/⟨PGDN⟩ for a special purpose (scrolling) by default.

In order to mark text (see Section 5.4 [General Keys], page 11) while scrolling pages, you can use the alternatives (see Section 5.3.2 [Alternative Movement Keys], page 10) ⟨SHIFT⟩-⟨CTRL⟩-⟨R⟩/⟨SHIFT⟩-⟨CTRL⟩-⟨C⟩.

If this annoys you, you might want to change your '`/etc/default.keytab`'. E.g., I moved scrolling to ⟨ALT⟩/⟨ALTGR⟩-⟨UP⟩/⟨DOWN⟩/⟨PGUP⟩/⟨PGDN⟩ and console switching to ⟨EXTRA⟩-*Fn* on my keyboard, so I can use ⟨SHIFT⟩-⟨PGUP⟩/⟨PGDN⟩ for marking pages and ⟨ALT⟩-*Fn* as hotkeys.

5.3.5 Notes About International Keyboards

The hotkeys ⟨ALT⟩-⟨-⟩ and ⟨ALT⟩-⟨]⟩ for the menu entries '`Search/Search`' (see Section 10.3.1 [Menu Search/Search], page 28) and '`Search/Replace`' (see Section 10.3.2 [Menu Search/Replace], page 28) are the positions on American keyboards. On international keyboards, they may vary. E.g., on German keyboards they are ⟨ALT⟩-⟨ß⟩ and ⟨ALT⟩-⟨+⟩. (Though ⟨ALT⟩-⟨-⟩ and ⟨ALT⟩-⟨]⟩ work, too, under Linux, at least with some keytabs.)

5.4 General Keys Used By PENG

⟨ESC⟩	abort most operations
⟨ENTER⟩	accept input or selection
⟨INS⟩	toggle insert/overwrite
⟨SHIFT⟩-cursor movement	mark text (input boxes and editors)
⟨SHIFT⟩-⟨DEL⟩	cut marked text (see Section 10.2.3 [Menu Edit/Cut], page 27)
⟨CTRL⟩-⟨INS⟩	copy marked text (see Section 10.2.4 [Menu Edit/Copy], page 28)
⟨SHIFT⟩-⟨INS⟩	paste text (see Section 10.2.5 [Menu Edit/Paste], page 28)
⟨CTRL⟩-⟨DEL⟩	delete marked text (see Section 10.2.6 [Menu Edit/Delete], page 28)
⟨ALT⟩-⟨SPACE⟩	non-breaking space

5.5 Keys Used In Menus

⟨LEFT⟩	move left in the main menu (with wrap-around)
⟨RIGHT⟩	move right in the main menu (with wrap-around)
⟨UP⟩	move up (with wrap-around)
⟨DOWN⟩	move down (with wrap-around) or open top-level submenu
⟨HOME⟩	move to the first position in the menu
⟨END⟩	move to the last position in the menu
⟨PGUP⟩	move up one screenful
⟨PGDN⟩	move down one screenful
⟨CTRL⟩-⟨LEFT⟩	move to the leftmost position in the menu
⟨CTRL⟩-⟨RIGHT⟩	move to the rightmost position in the menu
⟨CTRL⟩-⟨HOME⟩	move to the first position in the menu

CTRL-END	move to the last position in the menu
CTRL-PGUP	move to the top position in the menu
CTRL-PGDN	move to the bottom position in the menu
F10	go to the main menu
ALT-*A*	go to the top-level menu *A*
ENTER	select menu item or open submenu
DEL	remove item from the menu – some menus only, e.g., '`Window/List`' (see Section 10.8.6 [Menu Window/List], page 58), '`Window/Last windows`' (see Section 10.8.7 [Menu Window/Last windows], page 58, '`Window/Hotlist`' (see Section 10.8.8 [Menu Window/Hotlist], page 58), '`Options/Tools`' (see Section 10.7.5 [Menu Options/Tools], page 50) or '`Options/Variables`' (see Section 10.7.6 [Menu Options/Variables], page 51)
highlighted letter	directly select a menu item

Other hotkeys as listed in the menus.

5.6 Keys Used In String Input Boxes

LEFT	move to the left
RIGHT	move to the right, or when at the right end, copy characters from the current editor window, if any
HOME	move to the first position
END	move to the last position
DEL	delete character under the cursor
BACKSPACE	delete character left to the cursor
printable character	insert character
UP, DOWN	history (UP is compatible to many GNU programs; DOWN is Borland compatible – this, of course, is not to be construed as an assessment ;–)
CTRL-LEFT	jump one word to the left
CTRL-RIGHT	jump one word to the right
PGUP	scroll to the left
PGDN	scroll to the right
CTRL-T	delete to end of word and following spaces
CTRL-Y	clear input
TAB	file name completion (only for file input boxes); handles '`~`', '`~user`' and '`$ENVVAR`'
SHIFT-TAB	file name completion (in all input boxes)

File input boxes handle the usual globbing characters: '`*`', '`?`', '`[^-]`', '`{,}`', '`\`' escapes (don't work well under Dos, of course), and accept several file masks separated with spaces.

5.7 Keys Used In File Selection Boxes

LEFT	move left (with wrap-around to the previous line)
RIGHT	move right (with wrap-around to the next line)
UP	move up
DOWN	move down

'`/End`' tools are executed even if PENG is terminated by a '`TERM`' or '`HUP`' signal – see '`Options/Editor2/Use lock files`' (see Section 10.7.2.4 [Menu Options/Editor2/Use lock files], page 46). In this case, interactive tool functions, e.g., '`SELECT`' (see Section 11.6.24 [Tool SELECT], page 72), '`PROMPT`' (see Section 11.6.22 [Tool PROMPT], page 71) or '`YNQUESTION`' (see Section 11.6.23 [Tool YNQUESTION], page 72), are disabled, but the '`SHELL`' (see Section 11.6.30 [Tool SHELL], page 74) and '`STATUS`' functions (see Section 11.6.31 [Tool STATUS], page 74) *are* executed. Therefore, be careful about using them in '`/End`' tools – if the processes spawned don't terminate quickly enough, this might, e.g., prevent PENG from cleaning up properly before a system shutdown.

'`Command`' is the action for the tool (see Section 11.4 [Tool Evaluation], page 66).

'`Mask`', if given, can select which tool to apply, depending on the file name of the current editor window (e.g., to use different compilers for different languages with different file name extensions).

The '`Condition`', if not empty, is evaluated like a tool action when the tool is considered to be used, and should evaluate to '`True`' or '`False`' (case-insensitively). The tool is used if and only if the condition evaluates to '`True`' – otherwise other tools are considered to be used if their name, mask and condition are satisfied. This makes it possible to define tools dependent on, e.g., variables. Since conditions can be evaluated quite often, they should usually contain only *passive* functions like '`EQ`' (see Section 11.6.8 [Tool EQ], page 69) or '`NE`' (see Section 11.6.9 [Tool NE], page 69).

When tools have a mask and/or a condition, it does make sense to have several tools of the same menu name. They will be listed in the menu only once.

It is also possible to have different tools (reasonably with different masks and/or conditions) with the same hotkey, with the same or different menu names.

Under '`Options/Try a tool`' (see Section 10.7.7 [Menu Options/Try a tool], page 51), you can enter a tool command and apply it immediately without defining it as a tool.

The term *current editor window*, in the text above as well as in the description of the tool functions (see Section 11.6 [Tool Functions], page 67), refers to the currently active window, if any, unless that is the message window, in which case it refers to the window below it, if any.

11.3 Variables Set Automatically

When a tool is invoked, PENG sets the following environment variables (besides those defined as variables, see Section 11.1 [Defining Variables], page 63).

Some of them depend on the current editor window. Whenever a tool action influences the current editor window, the variables are updated automatically.

'`EDNAME`' (1)
: The file name of the current editor window. Empty if the current window is not an ordinary file, but something like an info reader, clipboard etc.

If the menu name is empty, the variable does not appear in any menu, but the environment variable will be set, anyway (see Section 11.1 [Internal Variables], page 64).

'`Info node`' can specify an info node (see Section 10.9.6 [Info Node Format], page 60) to be used as a context-sensitive help for this tool – see '`Help/Help`' (see Section 10.9.1 [Menu Help/Help], page 59). This makes it possible to distribute tools with documentation in separate info files without any changes to PENG. The built-in PENG tools use the same mechanism to refer to sections in this documentation.

11.1.0.1 Internally used tool variables

Tool variables can be used internally. E.g., the PENG built-in tools use an internal variable to compose the options to GPC from the various menu selections.

These variables are not normally listed in a menu because they only need to be changed when the tools themselves are modified to introduce new configuration options in the menu or things like that. However, you will see them under '`Options/Variables`' (see Section 10.7.6 [Menu Options/Variables], page 51).

11.2 Defining Tools

Tools can be defined under '`Options/Tools`' (see Section 10.7.5 [Menu Options/Tools], page 50).

'`Name`', '`Hotkey`' and '`Info node`' are the same as the menu name, hotkey and info node for variables (see Section 11.1 [Defining Variables], page 63). By convention, entries under '`Options`' should generally be used for variables only, but PENG does not enforce this.

As an additional feature, tools with the name '`/Begin`' or '`/End`' (note the leading '`/`') will not appear in any menu. '`/Begin`' tools are automatically executed when PENG is started – or, more precisely, when the config file containing them is loaded (when PENG is started or it is loaded explicitly with '`Options/Load config file`' (see Section 10.7.12 [Menu Options/Load config file], page 53). '`/End`' tools are automatically executed when PENG is terminated. This can be used, e.g., to create a temporary file or directory to be used by other tools at the beginning (see Section 11.6.21 [Tool TEMP], page 71), and automatically remove it at the end.

If you define several '`/Begin`' or '`/End`' tools, the order in which they are executed is not defined.

'`/End`' tools are executed after all windows have been closed. Therefore, tool functions that refer to open windows such as '`SAVE`' (see Section 11.6.39 [Tool SAVE], page 75), '`CLOSE`' (see Section 11.6.38 [Tool CLOSE], page 75), '`SEARCH`' (see Section 11.6.26 [Tool SEARCH], page 72), '`GOTO`' (see Section 11.6.40 [Tool GOTO], page 75), '`INSERT`' (see Section 11.6.41 [Tool INSERT], page 76) and '`#i`' output redirections (see Section 11.7 [Output Redirections], page 77) do nothing or cause an error, respectively. Furthermore, functions that open a window such as '`OPEN`' (see Section 11.6.35 [Tool OPEN], page 74), '`VIEW`' (see Section 11.6.36 [Tool VIEW], page 75), '`INFO`' (see Section 11.6.37 [Tool INFO], page 75), '`TRYINFO`' (see Section 11.6.29 [Tool TRYINFO], page 73) and '`#w`', '`#m`', '`#M`', '`#b`' and '`#B`' output redirections (see Section 11.7 [Output Redirections], page 77) are disabled.

⟨HOME⟩, ⟨CTRL⟩-⟨PGUP⟩	move to the first item
⟨END⟩, ⟨CTRL⟩-⟨PGDN⟩	move to the last item
⟨PGUP⟩	move up one screenful
⟨PGDN⟩	move down one screenful
⟨CTRL⟩-⟨LEFT⟩	move to the first column
⟨CTRL⟩-⟨RIGHT⟩	move to the last column
⟨CTRL⟩-⟨HOME⟩	move to the first line on the screen
⟨CTRL⟩-⟨END⟩	move to the last line on the screen
printable character	move to next file name starting with that character
⟨CTRL⟩-⟨A⟩ .. ⟨CTRL⟩-⟨Z⟩ (except ⟨CTRL⟩-⟨J⟩ and ⟨CTRL⟩-⟨M⟩)	go to a (user-definable) shortcut path (see Section 10.1.12 [Menu File/Shortcut paths], page 26)
⟨SHIFT⟩-⟨F4⟩	enter a new path

5.8 Keys Used In Color Selection Boxes

⟨LEFT⟩	move left (with wrap-around)
⟨RIGHT⟩	move right (with wrap-around)
⟨UP⟩	move up (with wrap-around)
⟨DOWN⟩	move down (with wrap-around)
⟨HOME⟩	move to the first item in the current line
⟨END⟩	move to the last item in the current line
⟨PGUP⟩, ⟨CTRL⟩-⟨PGUP⟩	move to the first line
⟨PGDN⟩, ⟨CTRL⟩-⟨PGDN⟩	move to the last line

5.9 Keys Used In The Text Editor

⟨LEFT⟩	move left
⟨RIGHT⟩	move right
⟨UP⟩	move up
⟨DOWN⟩	move down
⟨HOME⟩	move to the first position in the current line
⟨END⟩	move to the last position in the current line
⟨PGUP⟩	move up one screenful
⟨PGDN⟩	move down one screenful
⟨CTRL⟩-⟨PGUP⟩	move to beginning of text
⟨CTRL⟩-⟨PGDN⟩	move to ending of text
⟨CTRL⟩-⟨HOME⟩	move to the first line on the screen
⟨CTRL⟩-⟨END⟩	move to the last line on the screen
⟨CTRL⟩-⟨LEFT⟩	jump one word to the left
⟨CTRL⟩-⟨RIGHT⟩	jump one word to the right
⟨CTRL⟩-⟨UP⟩	jump to the previous position set by an '`INSERT`' tool (see Section 11.6.41 [Tool INSERT], page 76)
⟨CTRL⟩-⟨DOWN⟩	jump to the next position set by an '`INSERT`' tool
printable character	insert character
⟨DEL⟩	delete character under the cursor, but note '`Options/Editor1/Delete blocks`' (see Section 10.7.1.3 [Menu Options/Editor1/Delete blocks], page 43)
⟨BACKSPACE⟩	delete character before the cursor; unindent (not in paragraph mode), but note '`Options/Editor1/Delete blocks`' (see Section 10.7.1.3 [Menu Options/Editor1/Delete blocks], page 43)

⟨ENTER⟩	insert a newline if insert mode is on, otherwise move to next line
⟨TAB⟩	insert a tab character or do indentation, dependent on '`Options/Editor2/Use tabs`' (see Section 10.7.2.13 [Menu Options/Editor2/Use tabs], page 47)
⟨SHIFT⟩-⟨TAB⟩	file name completion
⟨CTRL⟩-⟨V⟩ control character	insert control character
⟨CTRL⟩-⟨T⟩	delete to end of word and following spaces
⟨CTRL⟩-⟨Y⟩	delete line
⟨CTRL⟩-⟨N⟩	insert line
⟨CTRL⟩-⟨W⟩	scroll the screen down without moving the cursor
⟨CTRL⟩-⟨Z⟩	scroll the screen up without moving the cursor
⟨CTRL⟩-⟨Q⟩⟨F⟩	search (see Section 10.3.1 [Menu Search/Search], page 28)
⟨CTRL⟩-⟨Q⟩⟨A⟩	replace (see Section 10.3.2 [Menu Search/Replace], page 28)
⟨CTRL⟩-⟨L⟩	search or replace again (see Section 10.3.3 [Menu Search/Search or replace again], page 29)
⟨CTRL⟩-⟨U⟩, ⟨CTRL⟩-⟨Q⟩⟨L⟩	undo (see Section 10.2.1 [Menu Edit/Undo], page 27)
⟨CTRL⟩-⟨Q⟩⟨R⟩	redo (see Section 10.2.2 [Menu Edit/Redo], page 27)
⟨CTRL⟩-⟨K⟩⟨B⟩	set the beginning of the block (see Section 10.7.1.2 [Menu Options/Editor1/Persistent blocks], page 43)
⟨CTRL⟩-⟨K⟩⟨K⟩	set the ending of the block (see Section 10.7.1.2 [Menu Options/Editor1/Persistent blocks], page 43)
⟨CTRL⟩-⟨K⟩⟨T⟩	mark current word
⟨CTRL⟩-⟨K⟩⟨H⟩	hide/unhide block
⟨CTRL⟩-⟨K⟩⟨C⟩	copy block
⟨CTRL⟩-⟨K⟩⟨V⟩	move block
⟨CTRL⟩-⟨K⟩⟨P⟩	print marked text/whole file (on Unix by using the '`lpr`' printer spooler, on Dos by writing to '`prn`')
⟨CTRL⟩-⟨K⟩⟨R⟩	insert from file
⟨CTRL⟩-⟨K⟩⟨W⟩	write marked text to file
⟨CTRL⟩-⟨K⟩⟨I⟩	indent marked text
⟨CTRL⟩-⟨K⟩⟨U⟩	unindent marked text
⟨CTRL⟩-⟨K⟩⟨E⟩	indent marked text from current column
⟨CTRL⟩-⟨K⟩⟨A⟩	unindent marked text from current column
⟨CTRL⟩-⟨K⟩⟨N⟩	convert marked text to lower case
⟨CTRL⟩-⟨K⟩⟨O⟩	convert marked text to upper case
⟨CTRL⟩-⟨K⟩⟨G⟩	insert the *quote string* into each marked line (see Section 10.7.1.7 [Menu Options/Editor1/Quote string], page 44)
⟨CTRL⟩-⟨K⟩⟨F⟩	insert the *mark string* into each marked line (see Section 10.7.1.8 [Menu Options/Editor1/Mark string], page 44)
⟨CTRL⟩-⟨K⟩⟨X⟩	remove first character from each marked line if it equals the first character of the first marked line
⟨CTRL⟩-⟨K⟩⟨Y⟩	delete marked text
⟨CTRL⟩-⟨K⟩⟨S⟩	save
⟨CTRL⟩-⟨K⟩⟨Q⟩, ⟨CTRL⟩-⟨K⟩⟨D⟩	leave the editor and go to the main menu
⟨CTRL⟩-⟨K⟩*n*	set label *n* (0 <= *n* <= 9)
⟨CTRL⟩-⟨Q⟩*n*	move to label *n* (0 <= *n* <= 9)
⟨CTRL⟩-⟨Q⟩⟨W⟩	jump backward to same indentation level
⟨CTRL⟩-⟨Q⟩⟨E⟩	jump forward to same indentation level
⟨CTRL⟩-⟨Q⟩⟨B⟩	jump to beginning of marked text
⟨CTRL⟩-⟨Q⟩⟨K⟩	jump to ending of marked text
⟨CTRL⟩-⟨Q⟩⟨S⟩	re-display message
⟨CTRL⟩-⟨Q⟩⟨C⟩	change case of the letter under the cursor

11 Tools

PENG supports extensive tool scripting capabilites. You can define your own tools, and use them via menus or hotkeys, and you can define environment variables that can be changed in the menus. This is described in the following sections.

11.1 Defining Variables

Variables can be defined under '`Options/Variables`' (see Section 10.7.6 [Menu Options/Variables], page 51). You can create new variables, delete existing ones (with ⟨DEL⟩), and edit existing ones.

'`Name`' is the name under which the variable appears in the menu, and '`Environment name`' is the name of the environment variable created. Furthermore, you can define a hotkey and some options that affects inputting the variable.

Please note: Some of the key combinations offered are already used for other purposes on some keyboards, e.g., ⟨ALTGR⟩+⟨2⟩/⟨3⟩/⟨7⟩/⟨8⟩/⟨9⟩/⟨0⟩/⟨Q⟩/⟨C⟩/⟨E⟩/⟨M⟩ on German keyboards, so you can't use them as hotkeys.

If '`Read from environment`' is selected and the environment variable is already set before starting PENG, the PENG variable is initialized from the environment variable's contents (except for enumeration variables). However, variable values stored in the config file (usually entered in previous PENG sessions) take precedence.

The values for enumeration variables are specified as a '`,`' separated list of identifiers which will appear in the menu, each one optionally followed by a '`:`' and a value to which the environment variable will be set (otherwise, the identifier is used for this purpose, too). '`\`' can be used to quote special characters ('`,`', '`:`' and itself).

Variables of string types (including filenames etc.) can be defined to be lists. Their value is a (possibly empty) list of such strings. The list can be edited as usual in a menu, with ⟨DEL⟩ to remove items (see Section 5.5 [Menu Keys], page 11). In the environment variable, the items are stored quoted appropriately for the usual shells and separated with spaces.

The menu name is of the general form '`[/`*mainmenu*`/][`*submenu*`/[`*subsubmenu*`/`...`]]`*menuentry*'.

It describes the menus and submenus for the variable. The only valid values for *mainmenu* are currently '`Run`', '`Compile`', '`Tools`', '`Help`' and '`Options`'. If omitted (including the leading '`/`'), it defaults to '`Tools`'.

The number and names of the submenus are not limited. (Yes, you can even use duplicate names within one menu, if that makes sense to you... ;–).

The letter to be used as a shortcut in each menu can be prefixed with a '`~`'. Otherwise, PENG will choose shortcuts by itself, as long as letters are available. A '`~`' at the end of the menu name means that no shortcut will be used at all.

⟨CTRL⟩-⟨Q⟩⟨Q⟩	reformat the current paragraph, *independently* of '`Options/Editor1/Paragraphs`' (see Section 10.7.1.6 [Menu Options/Editor1/Paragraphs], page 44)
⟨CTRL⟩-⟨Q⟩⟨P⟩	reformat all marked or all paragraphs, *independently* of '`Options/Editor1/Paragraphs`' (see Section 10.7.1.6 [Menu Options/Editor1/Paragraphs], page 44)
⟨CTRL⟩-⟨O⟩⟨S⟩	toggle simple format of status line/frame information (see Chapter 7 [Editor Status], page 19)
⟨CTRL⟩-⟨O⟩⟨D⟩	toggle Dos line endings
⟨CTRL⟩-⟨O⟩⟨R⟩	toggle read-only
⟨ESC⟩	leave the editor and go to the main menu, if '`Options/Editor2/Leave editor with Esc`' (see Section 10.7.2.6 [Menu Options/Editor2/Leave editor with Esc], page 46) is on

The key after ⟨CTRL⟩-⟨K⟩, ⟨CTRL⟩-⟨Q⟩, ⟨CTRL⟩-⟨O⟩ may or may not be combined with control.

If '`Options/Editor2/Multiple block operations`' (see Section 10.7.2.10 [Menu Options/Editor2/Multiple block operations], page 47) is enabled, you can do several ⟨CTRL⟩-⟨K⟩ block operations in a row without pressing ⟨CTRL⟩-⟨K⟩ again and again if you press ⟨CTRL⟩ with all the keys (e.g., ⟨CTRL⟩-⟨K⟩⟨I⟩⟨I⟩⟨I⟩⟨I⟩ to indent a block multiple columns). However, this does not work for

- ⟨CTRL⟩-⟨K⟩⟨0⟩ .. ⟨CTRL⟩-⟨K⟩⟨9⟩ (setting labels – quite pointless),
- ⟨CTRL⟩-⟨K⟩*x*⟨H⟩ (where *x* is any letter) because ⟨CTRL⟩-⟨H⟩ can't be distinguished from ⟨BACKSPACE⟩ and
- ⟨CTRL⟩-⟨K⟩*x*⟨K⟩ because the second ⟨CTRL⟩-⟨K⟩ is assumed to start a new block operation.

5.10 Keys Used In The Info Reader

The info reader generally understands all those keys used in the text editor (see Section 5.9 [Text Editor Keys], page 13) which do not modify the text, but the following keys have different meanings:

⟨TAB⟩	move to next link or reference
⟨SHIFT⟩-⟨TAB⟩	move to previous link or reference
⟨ENTER⟩	follow the link or reference under the cursor if any
⟨DEL⟩	enter a node to go to (see Section 10.9.6 [Info Node Format], page 60). Note that ⟨CTRL⟩-⟨G⟩ is an alternative for ⟨DEL⟩ (see Section 5.3.2 [Alternative Movement Keys], page 10).
⟨ESC⟩	close the info reader
⟨.⟩	follow the '`Next`' link if present
⟨,⟩	follow the '`Prev`' link if present
⟨<⟩	follow the '`Up`' link if present
other printable characters	move to the next link or reference starting with the characters entered. The characters are forgotten with every other movement.
⟨BACKSPACE⟩	forget the last character entered

Also, please note the '**Help/Last help**' (see Section 10.9.7 [Menu Help/Last help], page 60) function (hotkey: ⟨ALT⟩-⟨F1⟩).

5.11 Keys Used In The Message Window

⟨LEFT⟩	scroll the screen right
⟨RIGHT⟩	scroll the screen left
⟨HOME⟩	scroll the screen to the first position
⟨END⟩	scroll the screen to the last position
⟨UP⟩	move up
⟨DOWN⟩	move down
⟨PGUP⟩	move up one screenful
⟨PGDN⟩	move down one screenful
⟨CTRL⟩-⟨PGUP⟩	move to the first line
⟨CTRL⟩-⟨PGDN⟩	move to the last line
⟨CTRL⟩-⟨HOME⟩	move to the first line on the screen
⟨CTRL⟩-⟨END⟩	move to the last line on the screen
⟨ENTER⟩	seek to the position of the current message in an editor window
⟨ESC⟩	close the message window

If no info reader is opened, it re-opens the last info reader closed.

Finally, if no info reader has ever been open in this PENG session, it opens the top node of the PENG documentation in an info reader (see Section 10.9.2 [Menu Help/PENG Help], page 59).

10.9.8 The Menu Entry '`Help/ASCII table`'

Opens a read-only editor which contains a table of the character set used on the system (usually ASCII with extensions, e.g., ISO-8859-1). In particular, you can copy and paste special characters from this table which you may not be able to enter on your keyboard.

10.9.9 The Menu Entry '`Help/About`'

Shows the *About* box of PENG with copyright notice. There are no hidden 3D adventures in it. ;–)

10.9.10 The PENG Built-In Tool '`Help/Syntax help`'

This function tries to find help on the current word in the current editor.

Depending on the language (as determined by the file extension), it uses the following information sources (where *foo* stands for the current word):

Pascal: It tries to find the info node '`(gpc)` *foo*', i.e., the topic *foo* in the GPC documentation (see section "GNU Pascal" in *the GNU Pascal manual*). If this fails, it tries to find a man page for *foo*.

C: It searches for *foo* in the function index of the libc documentation (see section "Function Index" in *the libc manual*), and follows the link if found. If this fails, it tries to find a man page for *foo*.

Texinfo: It searches for *foo* in the command and variable index of the Texinfo documentation (see section "Command and Variable Index" in *the Texinfo manual*), and follows the link if found.

LaTeX: It searches for *foo* in the command index of the LaTeX documentation (see section "Command Index" in *the LaTeX manual*), and follows the link if found.

Other files:
It tries to find a man page for *foo*.

10.9.6 The Format Of An Info Node Description

An info node in PENG can be specified in the same format as accepted by other info readers (see section "Adding a new node to Info" in *the info manual*).

The general format is '`(`*filename*`)` *nodename*'.

filename, together with the parentheses, may be omitted. In this case, the node is assumed to be in the current file (or the PENG documentation when opening a new info reader).

If *filename* is given, is it searched for in the directories given in the '`INFOPATH`' (or '`INFODIR`' if '`INFOPATH`' is not set) environment variable, unless it contains an absolute path or a path starting with '`./`'.

The file extension '`.info`' is appended automatically if necessary. As an extension, PENG also recognizes the extension '`.inf`' which may have been used for info files created on filename-challenged operating systems.

Info nodes compressed by '`compress(1)`', '`gzip(1)`' or '`bzip2(1)`' are uncompressed automatically, provided the uncompressor ('`uncompress`', '`gunzip`' or '`bunzip2`', respectively) can be found in your path. File names are case-sensitive on most systems, but PENG will also try the lower case version of an info file name that contains any capital letters.

nodename specifies the name of the node in the file. Node names are sensitive to special characters including embedded whitespace, but case-insensitive.

nodename can be omitted. Then it defaults to '`Top`' which is by convention the main node in most info files.

If *nodename* is '`*`', it refers to the whole file given by *filename*.

PENG's info reader contains a caching mechanism which will keep info files in memory even for some time after they have been closed. This is done to reduce the time needed to go back to the info file, e.g., with '`Help/Last help`' (see Section 10.9.7 [Menu Help/Last help], page 60) – especially for compressed info files which will not have to be uncompressed again.

PENG will reload info files when they change. It will check for changes at the same times that text editors do, see '`Options/Editor2/Always check file time`' (see Section 10.7.2.8 [Menu Options/Editor2/Always check file time], page 47), and whenever you go to a new info node.

10.9.7 The Menu Entry '`Help/Last help`'

This function depends on the info readers currently open:

If the foreground window contains an info reader, it goes back to the previous info node. Multiple back-stepping in a row is possible (up to a maximum of currently 32 nodes).

If the foreground window does not contain an info reader, it brings the topmost info reader to the top.

6 Resources For The X11 Version Of PENG

PENG recognizes among others the following X11 resources:

'`lines`' the number of lines (default 24, minimum 2)
'`cols`' the number of columns (default 80, minimum 2)
'`normalFont`' the name of a fixed width font (default '`7x13`')
'`boldFont`' the name of a fixed width bold font (default '`7x13bold`')
'`cursorColor`' the color of the text cursor (default '`Red`')

Please note: The dimensions of '`normalFont`' and '`boldFont`' **must** be the same.

So, e.g., you could put the following lines into your '`~/.Xresources`', '`~/.Xdefaults`', '`~/.OWdefaults`' or whatever your window manager uses, to customize PENG. After changing them, don't forget to reload the resources with '`xrdb(1)`', or restart X11.

```
xpeng.lines: 30
xpeng.cols: 100
xpeng.normalFont: 9x15
xpeng.boldFont: 9x15bold
xpeng.cursorColor: White
```

add an entry that's already in the Hotlist, it will become the first entry of the Hotlist. This can be used to reorder the Hotlist.

10.9 The 'Help' Menu

Besides the items listed here, this menu may contain additional tools and variables defined with an '/Help' prefix (see Chapter 11 [Tools], page 63).

10.9.1 The Menu Entry 'Help/Help'

Opens the PENG documentation in an info reader. You're reading it right now. :–)

This function is context-sensitive, i.e., it will try to find the info node most relevant to your current action. Note that some kinds of actions (e.g., menu selections and input boxes) get aborted by this function. This is because it opens an info reader as a normal window so you can use it in the usual way, e.g., follow links in it.

About the keys that work in an info reader, see Section 5.10 [Info Keys], page 15.

10.9.2 The Menu Entry 'Help/PENG Help'

Opens the top node of the PENG documentation in an info reader, from which you can read the whole PENG documentation by going through the info menus.

About the keys that work in an info reader, see Section 5.10 [Info Keys], page 15.

10.9.3 The Menu Entry 'Help/Info'

Opens the top of the info tree, provided you got a 'dir' or 'dir.info' file installed in the info path.

This is the node '(dir)' in the info node format (see Section 10.9.6 [Info Node Format], page 60).

10.9.4 The Menu Entry 'Help/Info node'

Lets you enter an info node to be opened in an info reader (see Section 10.9.6 [Info Node Format], page 60).

10.9.5 The Menu Entry 'Help/Info file'

Opens the info file whose name you enter in an info reader. This is the same as entering the file name in parentheses in 'Help/Info node' (see Section 10.9.4 [Menu Help/Info node], page 59), see Section 10.9.6 [Info Node Format], page 60.

10.8.2 The Menu Entry 'Window/Messages'

Opens the message window, in which messages created by tools (see Chapter 11 [Tools], page 63) are shown, see the 'm' and following tool output redirections (see Section 11.7 [Output Redirections], page 77).

Note that tools will open the message window automatically when necessary, but you can use this function as a comfortable way to bring the message window back to the top, or even to re-open it after you have closed it. The messages in the message window will not be lost when you close it, until another tool redirects its output to the message window.

10.8.3 The Menu Entry 'Window/Next'

Brings the lowest window to the top.

10.8.4 The Menu Entry 'Window/Previous'

Puts the topmost window to the bottom.

10.8.5 The Menu Entry 'Window/Close'

Closes the current window. This is equivalent to 'File/Close' (see Section 10.1.7 [Menu File/Close], page 26) to suit different preferences. :–)

10.8.6 The Menu Entry 'Window/List'

Shows you a list of all currently open windows. You can select one of them to bring it to the top, and you can close windows with ⟨DEL⟩.

10.8.7 The Menu Entry 'Window/Last windows'

Lets you select a window from a list of windows that were recently closed.

10.8.8 The Menu Entry 'Window/Hotlist'

Lets you select a window from the Hotlist (see Section 10.8.9 [Menu Window/Add to Hotlist], page 58).

10.8.9 The Menu Entry 'Window/Add to Hotlist'

This allows you to keep a list of commonly opened files or other editors to be opened easily. After opening an editor in the usual way, you can add it to the Hotlist, and after that, open it easily from the Hotlist. You can remove items from the Hotlist with ⟨DEL⟩, as usual. When you

7 Status Information Of The Text Editor Windows

'^K' ⟨CTRL⟩-⟨K⟩ was pressed
'^Q' ⟨CTRL⟩-⟨Q⟩ was pressed
'^O' ⟨CTRL⟩-⟨O⟩ was pressed
'^k' continued block operation possible (see Section 10.7.2.10 [Menu Options/Editor2/Multiple block operations], page 47)
'*' text was modified since last saving
'R' editor is read-only
'D' text uses Dos line endings

Current line
Current column
Total lines (*)
Total bytes (*)
Number of marked lines and columns if any (*)
Current file name

(*) Not shown if simple format was selected with ⟨CTRL⟩-⟨O⟩⟨S⟩ (see Section 5.9 [Text Editor Keys], page 13).

system, in addition to the regular options (see Section 10.7.15.15 [Menu Options/Pascal/General options], page 56).

10.7.15.19 The PENG Built-In Tool Variable '`Options/Pascal/Options for cross compilers/Extra debug options`'

These are actually several menu entries which have the respective target system name (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54) appended to them.

Here you can specify command-line options which will be given to GPC for debugging cross-compilations (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53) for the respective target system, in addition to the regular debug options (see Section 10.7.15.13 [Menu Options/Pascal/Debug options], page 56).

10.7.15.20 The PENG Built-In Tool Variable '`Options/Pascal/Options for cross compilers/Extra release options`'

These are actually several menu entries which have the respective target system name (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54) appended to them.

Here you can specify command-line options which will be given to GPC for release cross-compilations (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53) for the respective target system, in addition to the regular release options (see Section 10.7.15.14 [Menu Options/Pascal/Release options], page 56).

10.7.16 The '`Options/C`' Submenu

10.7.16.1 The PENG Built-In Tool Variable '`Options/C/General options`'

Here you can specify command line options which will be given to GCC for any compilation.

The GCC options should probably be more elaborate, like the Pascal options, but perhaps some C programmer likes to do that...

10.8 The '`Window`' Menu

10.8.1 The Menu Entry '`Window/Redraw screen`'

Redraws PENG's screen. Useful if it got corrputed, e.g., by some console messages.

10.7.15.13 The PENG Built-In Tool Variable '`Options/Pascal/Debug options`'

Here you can specify any additional command line options which will be given to GPC for debug compilations (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53), such as a define '`-DDEBUG`' which can be tested in a program with '`{$ifdef DEBUG}`' to determine whether to do some additional sanity checks.

10.7.15.14 The PENG Built-In Tool Variable '`Options/Pascal/Release options`'

Here you can specify any additional command line options which will be given to GPC for release compilations (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53), such as optimization options (e.g., '`-O3`').

10.7.15.15 The PENG Built-In Tool Variable '`Options/Pascal/General options`'

Here you can specify command line options which will be given to GPC for any compilation. The '`--automake`' option will be given automatically, but any other options you want every compilation to use, such as '`--executable-file-name`' or some warning options, can be specified here.

10.7.15.16 The PENG Built-In Tool Variable '`Options/Pascal/Use GP`'

Here you can specify if you want to call GPC directly with the '`--automake`' and '`--executable-file-name`' options or the GP make utility. GP will replace GPC's '`--automake`' feature soon, and then, this option will disappear from the PENG tools.

10.7.15.17 The PENG Built-In Tool Variable '`Options/Pascal/Options for cross compilers/Executable path`'

These are actually several menu entries which have the respective target system name (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54) appended to them.

Here you can specify the directory where executables cross-compiled by GPC for the respective target will be stored rather than in the regular executable path (see Section 10.7.15.6 [Menu Options/Pascal/Executable path], page 54).

10.7.15.18 The PENG Built-In Tool Variable '`Options/Pascal/Options for cross compilers/Extra options`'

These are actually several menu entries which have the respective target system name (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54) appended to them.

Here you can specify command-line options which will be given to GPC for cross-compilations (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53) for the respective target

8 Command Line Arguments

PENG understands the following command-line options:

'`-h`', '`--help`'
: Output a short option summary and exit

'`-v`', '`--version`'
: Output version information and exit

'`-N`', '`--no-builtin-tools`'
: Suppress loading of the built-in tools.

'`-n`', '`--no-default-config`'
: Suppress loading of the default configuration files. There can be two default config files, a global one (loaded first) and a per-user one, loaded after the global one.

  They do not have to exist, but if they exist, unless this option is given, they are automatically loaded after the built-in tools (unless '`-N`' or '`--no-builtin-tools`' is given) and before any project specific config files specified with the '`-c`' or '`--config`' options (see below).

  The names of the default config files are '`/etc/peng.conf`' and '`~/.peng`' under Unix. Under Dos, they are '`peng.ini`', either in the directory '*$DJDIR*`\etc\`' (if DJGPP is installed) or in the directory where '`peng.exe`' is installed, and '`peng.cfg`', either in the directory '*$HOME*' (if you set the environment variable '`HOME`') or in the directory where '`peng.exe`' is installed.

'`-c` *config-file*', '`--config` *config-file*'
: Load *config-file* (after the built-in tools and/or default configuration files depending on the '`-N`', '`--no-builtin-tools`', '`-n`' or '`--no-default-config`' options – therefore it must be given *after* possible '`-N`', '`--no-builtin-tools`', '`-n`' or '`--no-default-config`' options). When the configuration is saved later, it will be written to the last configuration file loaded with this option or via the '`Options/Load config file`' (see Section 10.7.12 [Menu Options/Load config file], page 53) menu. Only items that were changed from the default, or from the previously loaded configuration file, are written, in order to keep the files small. This also makes it possible to store global options in a global file and project specific options in other files, so that changing global options only requires changing one file, not every project configuration file. The *desktop* (information about open windows, contents of input boxes and history lists) is also stored in the last configuration file – but note '`Options/Save minimal config`' (see Section 10.7.11 [Menu Options/Save minimal config], page 53).

'`-d`', '`--dump-tools`'
: Dump the built-in tools to standard output and exit

'`-R`', '`--read-only`'
: This option causes all following files on the command-line to be opened read-only – but not files opened from the '`File/Open`' menu (see Section 10.1.1 [Menu File/Open], page 25), so an alias like '`peng -R`' can be used as a file viewer.

`+`*line*[:*column*]
: Specify the position to move to in the next file or info reader given on the command-line (see below). This is used (line only) e.g., by '`less(1)`', so PENG can be used as the editor invoked by less by setting the environment variable '`EDITOR`' to '`peng`'.

'-i *node*', '--info *node*'
: Open an info reader for *node* (see Section 10.9.6 [Info Node Format], page 60). Note: Remember to quote parentheses on the command-line when necessary.

'-r *filename*', '--view *filename*'
: Open *filename* read-only.

'*filename*'
: Open *filename* read-write (unless the '-R' or '--read-only' option was given before).

Windows opened from the command line are not kept as opened windows in the config file, and therefore are not opened again automatically when running PENG again, but can be easily re-opened with 'Window/Last windows' (see Section 10.8.7 [Menu Window/Last windows], page 58). However, additional windows opened while running PENG, or windows closed and opened again are kept as opened windows. This is to prevent many windows from accumulating when using PENG often with command line arguments to quickly edit a few files.

tions/Pascal/Code generation], page 53) with a native compiler (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54). When compiling with a cross-compiler, the name of the target platform will be appended to this directory automatically. The directory will be created if necessary.

You can specify a different directory for compilations for release (see Section 10.7.15.8 [Menu Options/Pascal/Release unit destination path], page 55) so you can have the same units compiled, say, with and without optimization, and GPC will always use the right ones.

10.7.15.8 The PENG Built-In Tool Variable 'Options/Pascal/Release unit destination path'

Here you can select the directory in which GPC will store the compiled units and modules ('.o', '.gpi' and '.gpm' files) when compiling a release version (see Section 10.7.15.2 [Menu Options/Pascal/Code generation], page 53) with a native compiler (see Section 10.7.15.5 [Menu Options/Pascal/Target], page 54). When compiling with a cross-compiler, the name of the target platform will be appended to this directory automatically. The directory will be created if necessary.

You can specify a different directory for compilations for debugging (see Section 10.7.15.7 [Menu Options/Pascal/Debug unit destination path], page 54) so you can have the same units compiled, say, with and without optimization, and GPC will always use the right ones.

10.7.15.9 The PENG Built-In Tool Variable 'Options/Pascal/Unit path'

Here you can specify a list of directories, spearated by ':' (';' under Dos), where GPC will search for units used.

10.7.15.10 The PENG Built-In Tool Variable 'Options/Pascal/Object path'

Here you can specify a list of directories, spearated by ':' (';' under Dos), where GPC will search for object files and files in other languages (e.g., C or assembler) used.

10.7.15.11 The PENG Built-In Tool Variable 'Options/Pascal/Include path'

Here you can specify a list of directories, spearated by ':' (';' under Dos), where GPC will search for include files used.

10.7.15.12 The PENG Built-In Tool Variable 'Options/Pascal/Library path'

Here you can specify a list of directories, spearated by ':' (';' under Dos), where GPC will search for libraries used.

and Section 10.7.15.20 [Menu Options/Pascal/Options for cross compilers/Extra release options], page 57) will be given in addition to the general options (see Section 10.7.15.15 [Menu Options/Pascal/General options], page 56), and whether the debug or release unit destination path will be used (see Section 10.7.15.7 [Menu Options/Pascal/Debug unit destination path], page 54, and Section 10.7.15.8 [Menu Options/Pascal/Release unit destination path], page 55).

10.7.15.3 The PENG Built-In Tool Variable '`Options/Pascal/Linking`'

Here you can select if you want GPC to link the libraries used by your Pascal programs statically or dynamically.

Static linking is rarely recommendable, because it links *all* libraries, including e.g., '`libc`', statically which easily makes executables very large. Dynamic linking will, of course, link those libraries statically for which no dynamic versions exist. (In particular, under DJGPP, where dynamic linking is not implemented, this option will have no effect.)

10.7.15.4 The PENG Built-In Tool Variable '`Options/Pascal/Language dialect`'

Here you can select the dialect option given to GPC.

The default ('`GNU Pascal`', i.e., no special dialect option at all) should be OK for most purposes unless you want to make sure that your program conforms to the Standard Pascal or Extended Pascal standards, or you have a program written for BP which uses a lot of "dirty tricks".

10.7.15.5 The PENG Built-In Tool Variable '`Options/Pascal/Target`'

Here you can select the target platform for GPC.

The default ('`Native`') compiles for your current system, the other choices compile for particular systems. Of course, you have to have the proper cross-compiler installed. The list of systems given here is more or less arbitrary. You can easily add or remove some of them by modifying the tools (e.g., edit the PENG config file in a text editor and search for all occurrences of the name of one system, to easily find all places that need changes).

10.7.15.6 The PENG Built-In Tool Variable '`Options/Pascal/Executable path`'

Here you can select the directory in which GPC will store the executable created.

10.7.15.7 The PENG Built-In Tool Variable '`Options/Pascal/Debug unit destination path`'

Here you can select the directory in which GPC will store the compiled units and modules ('`.o`', '`.gpi`' and '`.gpm`' files) when compiling a debug version (see Section 10.7.15.2 [Menu Op-

9 Format Of The Configuration Files

Since the config files are text files, most parts of them should be obvious. String values are always enclosed in '`"`', and non-printable characters as well as '`=`' and '`"`' are encoded quoted-printable. Boolean values are stored as '`Y`' or '`N`', enumerated values are stored verbatim, except values of user-defined enumeration variables (see Section 11.1 [Defining Variables], page 63), which are stored as ordinal values, starting from 0.

Entries describing windows consist of:

- 4 integers reserved for future extensions,
- a boolean reserved for future extensions,
- name,
- kind,
- read-only,
- current column,
- current line,
- screen offset column,
- screen offset line,
- block beginning column (*),
- block beginning line (*),
- block ending column (*),
- block ending line (*).

(*) 0 if nothing marked.

Entries describing tools and variables (see Chapter 11 [Tools], page 63), consist of the same items as in the dialog boxes under '`Options/Tools`' (see Section 10.7.5 [Menu Options/Tools], page 50) and '`Options/Variables`' (see Section 10.7.6 [Menu Options/Variables], page 51), in the same order.

Because the config files are text files, you can edit them with any text editor (including PENG itself, of course), e.g., to copy some tools you have created or merge someone else's tools.

If '`Options/Editor2/Use lock files`' (see Section 10.7.2.4 [Menu Options/Editor2/Use lock files], page 46) is turned on, PENG will also create a lock file for the active config file to prevent problems when editing those as text files with PENG or vim.

However, PENG will not complain if it can't create the lock file for the default global or local config file because of lack of write permissions – the directory of the global default config file is usually not writable by a normal user, and some people like to have their home directory not writable to prevent "dot file pollution"...

If your home directory is writable, you will be warned if you start two PENG sessions using your local config file: the first one will create a lock file for it and the second one will notice the existing lock file and report it to you. This is intentional because the two instances of PENG might overwrite each other's settings.

So, in this case, you're better off using separate config files for the different PENG instances. You can keep common settings in your default config file (and change them by invoking PENG without arguments), and the different settings of the different instances (e.g., information about opened windows, project specific tools or other settings) in the separate config files.

10.7.11 The Menu Entry '`Options/Save minimal config`'

If this option is on, only the *permanent* config settings are stored in config files. Temporary values such as the contents of input boxes, history lists, and the list of last windows are not.

To reduce an existing config file to the minimum, turn this option on, quit and restart PENG (otherwise the temporary values are still in memory), and turn the option back off if you want.

10.7.12 The Menu Entry '`Options/Load config file`'

Loads a configuration file (see Chapter 9 [Config Files], page 23).

10.7.13 The Menu Entry '`Options/Save config file`'

Saves the configuration to the current config file (see Chapter 9 [Config Files], page 23). If '`Options/Auto save options`' (see Section 10.7.10 [Menu Options/Auto save options], page 52) is on, this also happens automatically on '`File/Save all`' (see Section 10.1.11 [Menu File/Save all], page 26), when autosave happens (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52), and when PENG is exited.

10.7.14 The Menu Entry '`Options/Save config file as`'

Saves the configuration to a new file (see Chapter 9 [Config Files], page 23).

10.7.15 The '`Options/Pascal`' Submenu

10.7.15.1 The PENG Built-In Tool '`Options/Pascal/Show sample GPC command line`'

This shows the command line that would be given to GPC when applying the '`Compile/Make`' (see Section 10.5.2 [Menu Compile/Make], page 33) tool to a Pascal file.

It is dependent on the current settings in the '`Options/Pascal`' menu.

10.7.15.2 The PENG Built-In Tool Variable '`Options/Pascal/Code generation`'

Here you can select if you want to compile a *draft* (debugging) or a final (release) version of your Pascal programs.

This does not directly pass any switches to GPC, but decides whether the debug or release options (see Section 10.7.15.13 [Menu Options/Pascal/Debug options], page 56, and Section 10.7.15.14 [Menu Options/Pascal/Release options], page 56; for cross-compilers see also Section 10.7.15.19 [Menu Options/Pascal/Options for cross compilers/Extra debug options], page 57,

10.7.8.6 The Menu Entry ‘Options/Message window/Maximum number of messages’

Here you can specify the maximum number of messages the message window will accept. When a tool redirected to the message window (see Section 11.7 [Output Redirections], page 77) produces more lines of output, it will be aborted.

This is independent of the size of the message window on the screen (see Section 10.7.8.1 [Menu Options/Message window/Message window size], page 51).

10.7.9 The ‘Options/Autosave’ Submenu

10.7.9.1 The Menu Entry ‘Options/Autosave/Autosave’

If this option is enabled, all modified files and configuration (see Section 10.1.11 [Menu File/Save all], page 26) will automatically be saved repeatedly either after specified time intervals – if the option is set to ‘Time’, see ‘Options/Autosave/Autosave seconds’ (see Section 10.7.9.2 [Menu Options/Autosave/Autosave seconds], page 52) – or after a specified number of keys pressed – if the option is set to ‘Keys’, see ‘Options/Autosave/Autosave keys’ (see Section 10.7.9.3 [Menu Options/Autosave/Autosave keys], page 52).

Please note: when unnamed files were modified, autosave will prompt for a file name to save. This can be irritating when it’s unexpected.

10.7.9.2 The Menu Entry ‘Options/Autosave/Autosave seconds’

If ‘Options/Autosave/Autosave’ (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52) is set to ‘Time’, you can specify here the length in seconds of the intervals after which autosave happens.

10.7.9.3 The Menu Entry ‘Options/Autosave/Autosave keys’

If ‘Options/Autosave/Autosave’ (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52) is set to ‘Keys’, you can specify here the number of key presses after which autosave happens.

10.7.10 The Menu Entry ‘Options/Auto save options’

If this option is on, the configuration will be saved (see Section 10.7.13 [Menu Options/Save config file], page 53) on ‘File/Save all’ (see Section 10.1.11 [Menu File/Save all], page 26), when autosave happens (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52), and when PENG is exited.

10 Menu Items

10.1 The ‘File’ Menu

10.1.1 The Menu Entry ‘File/Open’

This opens a file read-write. However, if you don’t have write permissions, it will be opened read-only automatically (see Section 10.1.3 [Menu File/View], page 25). If the file is already open, this will just open a new editor (see Section 10.1.6 [Menu File/New editor], page 26), without changing the read-write/read-only status.

If the file does not exist (but its directory exists), this function will open a new empty text editor which will be written to that file name when you save it (see Section 10.1.9 [Menu File/Save], page 26).

10.1.2 The Menu Entry ‘File/Open file at cursor’

This works like ‘File/Open’ (see Section 10.1.1 [Menu File/Open], page 25), but copies the word under the cursor from the current editor window as the initial content and allows you to copy characters from the current editor window.

10.1.3 The Menu Entry ‘File/View’

This opens a file read-only. Read-only files can be viewed and copied from with the usual editor keys, but they cannot be modified. They can be written to a new file name, however (see Section 10.1.10 [Menu File/Save as], page 26). If the file is already open, this will just create a new editor (see Section 10.1.6 [Menu File/New editor], page 26), without changing the read-write/read-only status.

10.1.4 The Menu Entry ‘File/View file at cursor’

This works like ‘File/View’ (see Section 10.1.3 [Menu File/View], page 25), but copies the word under the cursor from the current editor window as the initial content and allows you to copy characters from the current editor window.

10.1.5 The Menu Entry ‘File/New file’

This opens a new (read-write) unnamed text editor. When you try to save it (see Section 10.1.9 [Menu File/Save], page 26), you will be prompted for a file name.

If you want to give the filename immediately, you can do this with ‘File/Open’ (see Section 10.1.1 [Menu File/Open], page 25) instaed.

10.1.6 The Menu Entry '`File/New editor`'

Opens a new editor for the file in the current editor. The new editor will start at the same position as the current one (so one might be misled into thinking the function didn't work...). Several editors for one file will be marked by '`:`*n*', with consecutive numbers *n*.

10.1.7 The Menu Entry '`File/Close`'

Closes the current window. This is equivalent to '`Window/Close`' (see Section 10.8.5 [Menu Window/Close], page 58) to suit different preferences. :–)

10.1.8 The Menu Entry '`File/Change directory`'

Changes the current working directory for PENG as well as OS shells (see Section 10.1.13 [Menu File/OS shell], page 26) and tools (see Chapter 11 [Tools], page 63) subsequently spawned by PENG.

10.1.9 The Menu Entry '`File/Save`'

Saves the current editor. If it is read-only, it will ask for a new file name to save it to, just like '`File/Save as`' (see Section 10.1.10 [Menu File/Save as], page 26).

10.1.10 The Menu Entry '`File/Save as`'

Saves the current editor under a new file name. This also makes it possible to save read-only files and non-file windows, e.g., the message window or info readers, to a file.

10.1.11 The Menu Entry '`File/Save all`'

Saves all editor windows which contain modified changes. If '`Options/Auto save options`' (see Section 10.7.10 [Menu Options/Auto save options], page 52) is on, it also saves the configuration to the current config file.

10.1.12 The Menu Entry '`File/Shortcut paths`'

In this menu, you can enter some path shortcuts which you can easily access in file selection boxes (see Section 5.7 [File Selection Keys], page 12).

10.1.13 The Menu Entry '`File/OS shell`'

Invokes an OS shell and resumes to PENG when you leave the shell (in most shells, this is done by entering '`exit`'). The name of the shell is taken from the usual environment variable containing it ('`SHELL`' under Unix, '`COMSPEC`' under Dos).

10.7.6 The Menu Entry '`Options/Variables`'

Lets you define, modify or remove variables (see Section 11.1 [Defining Variables], page 63).

10.7.7 The Menu Entry '`Options/Try a tool`'

Lets you enter a tool command and apply it immediately without defining it as a tool (see Chapter 11 [Tools], page 63).

10.7.8 The '`Options/Message window`' Submenu

10.7.8.1 The Menu Entry '`Options/Message window/Message window size`'

Specifies the number of lines the message window will occupy on the screen. This is independent of the number of messages it will accept (see Section 10.7.8.6 [Menu Options/Message window/Maximum number of messages], page 52).

10.7.8.2 The Menu Entry '`Options/Message window/Message window at top`'

Specifies if the message window will appear at the top or bottom of the screen.

10.7.8.3 The Menu Entry '`Options/Message window/Ignore files`'

Files whose names match this mask will not be opened when messages refer to them. Useful for names of binary files, e.g., object files and libraries and the linker itself, which may appear in linker messages, but should not be loaded into an editor.

10.7.8.4 The Menu Entry '`Options/Message window/Automatically go to first message`'

If this option is on, tools (see Chapter 11 [Tools], page 63) with an '`M`' or '`B`' output redirection (see Section 11.7 [Output Redirections], page 77) will automatically seek to the position of the first message in an editor window, so it can be edited immediately.

10.7.8.5 The Menu Entry '`Options/Message window/Skip messages without line number`'

If this option is on, movements in the message window (see Section 5.11 [Message Window Keys], page 16) will skip over messages without line numbers (which are usually comments). Those messages are still shown, of course, but this way you can go to the messages with line numbers more easily. An exception is only when you move beyond the first or last message with a line number.

10.7.3.4 The Menu Entry '`Options/Environment/Control-Z to suspend`'

If this option is enabled, CTRL-Z will suspend PENG, just like '`File/Suspend`' (see Section 10.1.14 [Menu File/Suspend], page 27) does.

10.7.3.5 The Menu Entry '`Options/Environment/Control-C to exit`'

If this option is enabled, CTRL-C will quit PENG, just like '`File/Exit`' (see Section 10.1.15 [Menu File/Exit], page 27) does.

10.7.3.6 The Menu Entry '`Options/Environment/Simulate block cursor`'

In editors and input boxes, PENG indicates insert mode by a block cursor, but not all terminals support this. If this option is active, PENG will simulate the block cursor with graphic characters.

10.7.4 The '`Options/Colors`' Submenu

In these submenus, you can edit all the colors used by PENG. You can specify separate colors for color terminals, X11 and monochrome terminals. The reason there are distinct colors for X11 is that on X11 often dark colors on light background are preferred, while on color terminals often light colors on dark background are preferred.

The first two lines in a color selection box can be used to select the foreground and background color, respectively, on a color terminal. The next two lines correspond to the color on X11. The last line offers some attributes for monochrome terminals (not all of which are available on all terminals, of course). To the left of the three fields, there is a sample character. For a list of keys used in the color selection boxes, see Section 5.8 [Color Selection Keys], page 13.

Most color definitions are optional. This is indicated by a hyphen ('`-`') on the left side of the color selection box. If no color is selected (i.e., the hyphen is selected), the next appropriate color will be used. E.g., the color of the main menu items defaults to that of normal menu items. It is possible to select a color only for one of foreground or background, and let the other one be determined by the next appropriate color. This is done a lot in the default settings, so you can easily change, e.g., the background color of the editor, by changing it under '`Normal text`' without having to change it under many items (but, of course, you can still select different backgrounds for the other items if you like).

Some optional colors use inverted versions of other colors (e.g., the inverted color of normal text for the color for marked text), and a few optional colors are handled specially, e.g., the foreground color for tab characters in the text editor. If this is not selected, tab characters are displayed as control characters, i.e., as '`I`' (because tab is ASCII #9 which is '`Control-I`') in the color of control characters. If that is not selected, all control characters will be invisible. The same holds for non-breaking spaces.

10.7.5 The Menu Entry '`Options/Tools`'

Lets you define, modify or remove tools (see Section 11.2 [Defining Tools], page 64).

10.1.14 The Menu Entry '`File/Suspend`'

Suspends PENG if your system and shell support job control (just like CTRL-Z does with most programs).

You can also use CTRL-Z to suspend PENG if '`Options/Environment/Control-Z to suspend`' (see Section 10.7.3.4 [Menu Options/Environment/Control-Z to suspend], page 50) is enabled.

If any editor window contains unsaved changes, PENG will save them automatically if autosave is active (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52).

PENG can be continued with the normal continue command of your shell, usually called '`fg`'.

10.1.15 The Menu Entry '`File/Exit`'

Exits PENG. If any editor window contains unsaved changes, PENG will save them automatically if autosave is active (see Section 10.7.9.1 [Menu Options/Autosave/Autosave], page 52), and otherwise ask you whether you want to save them.

You can also use CTRL-C to exit PENG if '`Options/Environment/Control-C to exit`' (see Section 10.7.3.5 [Menu Options/Environment/Control-C to exit], page 50) is enabled.

10.2 The '`Edit`' Menu

10.2.1 The Menu Entry '`Edit/Undo`'

Undoes the effect of the last operation(s) in an editor. Several operations of the same kind (movement, insertion, deletion) are undone by a single undo. It is possible to use undo multiple times (up to 128 times currently) in a row.

An operation undone inadvertently can be redone (see Section 10.2.2 [Menu Edit/Redo], page 27), but only **immediately** after the undo.

Please note: undos may not be correct if the same file is opened in several windows, and changes are made in one window between an operation and the corresponding undo in another window.

10.2.2 The Menu Entry '`Edit/Redo`'

Redoes an operation that was undone in an editor (see Section 10.2.1 [Menu Edit/Undo], page 27). It works only **immediately** after the undo.

10.2.3 The Menu Entry '`Edit/Cut`'

Cuts the marked text from a text editor or an input box to the clipboard.

10.2.4 The Menu Entry 'Edit/Copy'

Copies the marked text from a text editor or an input box to the clipboard, without removing it from the text editor or input box.

10.2.5 The Menu Entry 'Edit/Paste'

Pastes the active text in the clipboard to a text editor or an input box.

10.2.6 The Menu Entry 'Edit/Delete'

Deletes the marked text from a text editor or an input box, without copying it to the clipboard.

10.2.7 The Menu Entry 'Edit/Edit'

This brings you to the topmost window. Pressing ESC while in the main menu or a top-level submenu does the same. The hotkey ALT-0 can be used to get from several menu levels or input boxes to the window faster than by repeatedly pressing ESC.

10.2.8 The Menu Entry 'Edit/Open clipboard'

Opens the clipboard so you can edit it. Text cut or copied from elsewhere is always appended at the end of the clipboard, but you can mark text anywhere within the clipboard to select it for following paste operations.

10.2.9 The Menu Entry 'Edit/Clear clipboard'

Clears the clipboard. Especially useful if you store the clipboard in a permanent file (see Section 10.7.2.3 [Menu Options/Editor2/Clipboard file], page 45).

10.3 The 'Search' Menu

10.3.1 The Menu Entry 'Search/Search'

Finds an occurrence of a string or regular expression in the current editor, depending on various options in the 'Search' menu (see Section 10.3 [Menu Search], page 28).

10.3.2 The Menu Entry 'Search/Replace'

Replaces one or all occurrences of a string or regular expression in the current editor by another string, depending on various options in the 'Search' menu (see Section 10.3 [Menu Search], page 28).

10.7.2.19 The Menu Entry 'Options/Editor2/Line number width'

This determines the width that line numbers are displayed when 'Options/Editor1/Line numbers' (see Section 10.7.1.10 [Menu Options/Editor1/Line numbers], page 45) is enabled. Line numbers longer than the width given here will be truncated, but marked with a left arrow to avoid confusion.

10.7.2.20 The Menu Entry 'Options/Editor2/Lexical highlighting'

Turns lexical highlighting in text editor windows on or off. Note that currently lexical highlighting is quite primitive.

The colors used for highlighting as well as the keywords recognized can be specified under 'Options/Colors/Lex' (see Section 10.7.4 [Menu Options/Colors], page 50).

10.7.3 The 'Options/Environment' Submenu

10.7.3.1 The Menu Entry 'Options/Environment/Frame windows'

If this option is on, all editor and info reader windows get a frame around them, and their status information is written onto the frame rather than into the top line.

10.7.3.2 The Menu Entry 'Options/Environment/Screen size'

Shows the current screen size.

Under certain circumstances, it also allows you to change the screen size between 80x25 and 80x50 characters. Only implemented for DJGPP and local Linux consoles. For Linux consoles, it requires setfont to be installed and set up properly and to have the necessary permissions; if your current screen size is not 80x25 or 80x50, this might not work or produce unexpected screen sizes. If you prefer using other tools like 'SVGATextMode(8)', you can call them on the command line, or as a PENG tool (see Chapter 11 [Tools], page 63). PENG will react to the changed screen size automatically. On Linux consoles, resizing will affect all virtual consoles, not just the current one.

Under X11, the window size can't be changed with this menu item, but you can simply resize your window with the mouse (and however else your window manager may support it). On how to determine the size at start-up, please see Chapter 6 [Resources], page 17.

10.7.3.3 The Menu Entry 'Options/Environment/Clock'

Here you can choose if you want the current time, date, or both to be shown at the top right corner of PENG's screen.

Otherwise it will indent the current line by filling with spaces up to the position after the next space in the previous (non-empty) line, and if there are no more spaces, insert the number of spaces given under '`Options/Editor2/Indent width`' (see Section 10.7.2.16 [Menu Options/Editor2/Indent width], page 48).

Note that even when this option is off, you can get a tab character by pressing ⟨CTRL⟩-⟨V⟩ ⟨TAB⟩.

10.7.2.14 The Menu Entry '`Options/Editor2/Write with tabs`'

If this option is active, PENG will replace large amounts of leading spaces in a line by tab characters when saving text editor files. The number of columns where the tab positions are assumed to be can be given under '`Options/Editor2/Tab size`' (see Section 10.7.2.15 [Menu Options/Editor2/Tab size], page 48).

10.7.2.15 The Menu Entry '`Options/Editor2/Tab size`'

Here you can enter the tab size which is used for expanding tabs in text editor windows (see Section 10.7.2.12 [Menu Options/Editor2/Expand tabs], page 47) and info files (which is done automatically) and for writing files with tabs (see Section 10.7.2.14 [Menu Options/Editor2/Write with tabs], page 48). The value is also set to the variable '`TABSIZE`' (see Section 11.3 [Automatic Variables], page 65) when invoking PENG tools (see Chapter 11 [Tools], page 63).

10.7.2.16 The Menu Entry '`Options/Editor2/Indent width`'

If '`Options/Editor2/Use tabs`' (see Section 10.7.2.13 [Menu Options/Editor2/Use tabs], page 47) is off, and there are no more spaces in the previous (non-empty) line, pressing ⟨TAB⟩ in a text editor will insert the number of spaces given here.

It also determines the number of spaces '`\I`' will expand to in an '`INSERT`' tool function (see Section 11.6.41 [Tool INSERT], page 76), and, therefore, the indentation the syntax macros among the built-in PENG tools (see Section 11.6.41.1 [Syntax Macros], page 76) will use.

Set it to your favourite indentation depth.

10.7.2.17 The Menu Entry '`Options/Editor2/Paragraphs in files`'

Since the paragraph mode (see Section 10.7.1.6 [Menu Options/Editor1/Paragraphs], page 44) has a strong effect on editing, it can be enabled for only some kinds of files, e.g., for mail files, but not for associated data or program files that are often accessed while editing mail. Every file whose name (without directory, of course) matches the mask specified here will use the paragraph mode, if it is enabled at all.

10.7.2.18 The Menu Entry '`Options/Editor2/Line width`'

This determines the length of lines in paragraphs (see Section 10.7.1.6 [Menu Options/Editor1/Paragraphs], page 44).

C-like escape sequences (in particular '`\n`', but **not** '`\007`' because of conflicts with the backreference syntax, see below) are recognized in the replacement string.

If '`Search/Regular expression`' (see Section 10.3.6 [Menu Search/Regular expression], page 29) is enabled, the replacement string may also contain *backreferences*, i.e., it can reference

- the whole matched expression with '`&`' or '`\0`',
- the 7'th subexpression with '`\7`' (just like in regular expressions, but using it in replacements is not slow :–),
- the 7'th subexpression converted to upper/lower case with '`\u7`' or '`\l7`', respectively,
- the whole matched expression converted to upper/lower case with '`\u0`' or '`\l0`', of course, and
- a verbatim '`&`' or '`\`' with '`\&`' or '`\\`', respectively.

10.3.3 The Menu Entry '`Search/Search or replace again`'

Repeats the last search or replace operation. The new operation will always start after the current cursor position, regardless of the setting of '`Search/From cursor`' (see Section 10.3.9 [Menu Search/From cursor], page 31).

10.3.4 The Menu Entry '`Search/Case sensitive`'

Besides the usual two options ('`Never`' and '`Always`') there is the option '`Smart`' which causes a search to be case sensitive if and only if the search string contains any upper case letter (just like the '`-i`' option to '`less(1)`').

10.3.5 The Menu Entry '`Search/Whole words only`'

If this option is on, search (see Section 10.3.1 [Menu Search/Search], page 28) and replace (see Section 10.3.2 [Menu Search/Replace], page 28) operations will only find whole words, consisting of letters, digits and underscores ('`_`').

10.3.6 The Menu Entry '`Search/Regular expression`'

If this is enabled, you can search for regular expressions, and replace them using subexpression references. The difference between basic and extended regular expressions (see Section 10.3.7 [Menu Search/Extended regex], page 31) is not functionality, but only syntax. These concepts are explained more extensively in the man pages of programs like '`grep(1)`' and '`sed(1)`', so here's only a short reference on regular expressions:

Basic	Extended	Meaning
'`.`'	'`.`'	matches any single character
'`[aei-z]`'	'`[aei-z]`'	matches either '`a`', '`e`', or any character from '`i`' to '`z`'
'`[^aei-z]`'	'`[^aei-z]`'	matches any character but '`a`', '`e`', or '`i`' .. '`z`' To include in such a list the characters '`]`', '`^`', or '`-`', put them first, anywhere but first, or first or last, respectively.

<table>
<tr><td>‘[[:alnum:]]’</td><td>‘[[:alnum:]]’</td><td>matches any alphanumeric character</td></tr>
<tr><td>‘[^[:digit:]]’</td><td>‘[^[:digit:]]’</td><td>matches anything but a digit</td></tr>
<tr><td>‘[a[:space:]]’</td><td>‘[a[:space:]]’</td><td>matches the letter ‘a’ or a space character (space, tab, ...)</td></tr>
<tr><td>...</td><td></td><td>(there are more classes available)</td></tr>
<tr><td>‘\w’</td><td>‘\w’</td><td>= ‘[[:alnum:]]’</td></tr>
<tr><td>‘\W’</td><td>‘\W’</td><td>= ‘[^[:alnum:]]’</td></tr>
<tr><td>‘^’</td><td>‘^’</td><td>matches the empty string at the beginning of a line</td></tr>
<tr><td>‘$’</td><td>‘$’</td><td>matches the empty string at the end of a line</td></tr>
<tr><td>‘*’</td><td>‘*’</td><td>matches zero or more occurences of the preceding expression</td></tr>
<tr><td>‘\+’</td><td>‘+’</td><td>matches one or more occurences of the preceding expression</td></tr>
<tr><td>‘\?’</td><td>‘?’</td><td>matches zero or one occurence of the preceding expression</td></tr>
<tr><td>‘\{n\}’</td><td>‘{n}’</td><td>matches exactly n occurences of the preceding expression (n is an integer number)</td></tr>
<tr><td>‘\{m,n\}’</td><td>‘{m,n}’</td><td>matches m to n occurences of the preceding expression (m and n are integer numbers, m <= n)</td></tr>
<tr><td>‘a b’</td><td>‘a b’</td><td>matches a followed by b (a and b are regular expressions)</td></tr>
<tr><td>‘a\|b’</td><td>‘a|b’</td><td>matches a or b (a and b are regular expressions)</td></tr>
<tr><td>‘\(\)’</td><td>‘( )’</td><td>forms a subexpression, to override precedence, and for subexpression references</td></tr>
<tr><td>‘\7’</td><td>‘\7’</td><td>matches the 7’th subexpression (counted by their start in the regex), where 7 is a number from 1 to 9 ;–). Please note: using this feature can be very slow or take very much memory (exponential time and space in the worst case, if you know what that means...).</td></tr>
<tr><td>‘\’</td><td>‘\’</td><td>quotes the following character if it’s special (i.e., listed above)</td></tr>
<tr><td>rest</td><td>rest</td><td>any other character matches itself</td></tr>
</table>

Precedence, from highest to lowest:

– parentheses (‘\(\)’ or ‘( )’)
– repetition (‘*’, ‘\+’ or ‘+’, ‘\?’ or ‘?’, ‘\{ \}’ or ‘{ }’)
– concatenation
– alternation (‘\|’ or ‘|’)

Regular expression matches follow the principle *leftmost longest*, i.e., if there are several possible matches, they will find the one that starts leftmost, and if there are several ones starting at the same leftmost position, the longest one of those. The length is particularly important when replacing. Of course, continued searches will also find matches further to the right, but they will *not* find shorter ones starting at the same position. E.g., if you search for ‘f.*o’ in the text ‘foofo’, the search will first find the whole text (‘foofo’), and then the right ‘fo’, but never the left ‘foo’.

Please note: when you’re surprised about a regex search not finding what you expect, please check (besides the regex syntax, of course) the setting of ‘Search/Whole words only’ (see Sec-

10.7.2.8 The Menu Entry ‘Options/Editor2/Always check file time’

If this option is on, PENG checks the modification time of the current file after each key pressed, and offers to reload the file if it was modified. This gives additional security against accidentally editing a file that was already modified by other means (see Section 10.7.2.4 [Menu Options/Editor2/Use lock files], page 46), but can be time consuming on slower systems or when editing files on NFS mounted file systems. If the option is off, PENG only checks the time after selecting a function from a menu and at similar occasions.

If the option is on, PENG will also check the modification time of info files after each key pressed and, if modified, reload them without further request.

10.7.2.9 The Menu Entry ‘Options/Editor2/Move behind paste’

If this option is on, the cursor will be placed behind the inserted text after each insertion into a text editor, e.g., with ‘Edit/Paste’ (see Section 10.2.5 [Menu Edit/Paste], page 28). If it is off, it will be placed before it.

10.7.2.10 The Menu Entry ‘Options/Editor2/Multiple block operations’

If this option is on, the text editor allows multiple block operations to be done in a row without pressing CTRL-K every time (see Section 5.9 [Text Editor Keys], page 13).

10.7.2.11 The Menu Entry ‘Options/Editor2/Trim trailing spaces’

If this option in on, trailing spaces at the end of a line will be removed automatically as soon as the line is modified in any way.

Please note: Undo operations (see Section 10.2.1 [Menu Edit/Undo], page 27) do **not** restore those spaces.

10.7.2.12 The Menu Entry ‘Options/Editor2/Expand tabs’

If this option is active, tab characters (ASCII #9) in files opened in a text editor will automatically be replaced by an appropriate amount of spaces. The number of columns where the tab positions are assumed to be can be given under ‘Options/Editor2/Tab size’ (see Section 10.7.2.15 [Menu Options/Editor2/Tab size], page 48).

Please note: Expanding tabs can break files where tab characters are significant, in particular makefiles, so be careful with this option!

10.7.2.13 The Menu Entry ‘Options/Editor2/Use tabs’

If this option is active, the TAB key pressed in a text editor (see Section 5.9 [Text Editor Keys], page 13) will insert a tab character (ASCII #9).

10.7.2.4 The Menu Entry ‘Options/Editor2/Use lock files’

If this option is enabled, PENG will create a lock file for every file opened read-write, and if the lock file exists already, it won’t open the file without confirmation.

This is to prevent two instances of PENG from editing the same file at the same time. The name of the lock file is the name of the file opened plus ‘.swp’, like in ‘vim(1)’ – but note ‘Options/Editor2/Lock files with leading dot’ (see Section 10.7.2.5 [Menu Options/Editor2/Lock files with leading dot], page 46). Therefore, it also prevents PENG and ‘vim’ from editing the same file at the same time.

Unlike ‘vim’, PENG does not use the lock file to store any swap data, but maybe this will change. Currently, the only data stored in the lock file is a ‘vim’ compatible header block, so ‘vim’ will recognize PENG as the current editor of the file.

However, if PENG is terminated by a ‘TERM’ or ‘HUP’ signal (e.g., on system shutdown) while a file that has a lock file contains unsaved changes, a copy of the changed file is written into the lock file. Though PENG does not have a recovery function, it’s possible to extract the modified file manually in emergency cases by just removing the header (the first 4096 bytes of the file). Editor windows that don’t have a lock file are written without a header to a file with a temporary file name, usually in a temp directory. Since Dos can’t handle the lock file name in most cases (and Dos programs usually don’t get a ‘SIGHUP’ or ‘SIGTERM’ when the system is shut down ;–), this option is off by default in Dos.

10.7.2.5 The Menu Entry ‘Options/Editor2/Lock files with leading dot’

If this option is on, the lock file name (see Section 10.7.2.4 [Menu Options/Editor2/Use lock files], page 46) is preceded by a dot. That’s what newer versions of ‘vim’ (since 5.0 or so) do, so this option is on by default.

10.7.2.6 The Menu Entry ‘Options/Editor2/Leave editor with Esc’

If this option is on, you can go from a normal text editor window to the main menu with ⟨ESC⟩. This option is on by default for the ease of first-time PENG users and users on terminals with limited function keys.

Please note: Info readers and message windows will always be closed with ⟨ESC⟩, independently of this option.

10.7.2.7 The Menu Entry ‘Options/Editor2/Delete empty files’

If this option is on, whenever a text editor (more precisely, the last text editor opened on a given file) is *closed* whose contents are completely empty, the file on the disk is removed. If the option is set to ‘Ask’, PENG will ask whether to remove it.

tion 10.3.5 [Menu Search/Whole words only], page 29). This switch is respected in regex searches as well, but might be confusing sometimes.

10.3.7 The Menu Entry ‘Search/Extended regex’

This option selects between basic and extended regular expressions if regular expressions are enabled for searching (see Section 10.3.6 [Menu Search/Regular expression], page 29).

10.3.8 The Menu Entry ‘Search/Backwards’

If this option is on, search and replace operations will go through the text in backward direction, from the ending towards the beginning – but note ‘Search/From cursor’ (see Section 10.3.9 [Menu Search/From cursor], page 31).

10.3.9 The Menu Entry ‘Search/From cursor’

If this option is on, search and replace operations will only consider the text from the current cursor position to the end of the file – or to the beginning if ‘Search/Backwards’ (see Section 10.3.8 [Menu Search/Backwards], page 31) is on. Otherwise, they will start at the beginning (ending) of the file. But note ‘Search/Marked text only’ (see Section 10.3.10 [Menu Search/Marked text only], page 31).

10.3.10 The Menu Entry ‘Search/Marked text only’

If this option is on, PENG will only consider marked text for search and replace operations, whenever some text is marked (otherwise the whole file).

This is in addition to restrictions set by ‘Search/From cursor’ (see Section 10.3.9 [Menu Search/From cursor], page 31).

10.3.11 The Menu Entry ‘Search/Prompt on replace’

If this option is on, PENG will ask for confirmation before each replacement caused by ‘Search/Replace’ (see Section 10.3.2 [Menu Search/Replace], page 28). An answer of ‘y’ will do the replacement, ‘n’ will skip it, and ⟨ESC⟩ will abort the replacement operation altogether.

If the replacement string contains back-references to regular expressions (see Section 10.3.6 [Menu Search/Regular expression], page 29), PENG will also show the actual replacement text for each possible replacement while asking for confirmation.

10.3.12 The Menu Entry ‘Search/Replace all’

If this option is on, replacement operations (see Section 10.3.2 [Menu Search/Replace], page 28) will only consider the first match found, otherwise every match found.

10.3.13 The Menu Entry '`Search/Go to line`'

This lets you enter a line number to go to in the current file.

10.3.14 The Menu Entry '`Search/Replace identifier`'

This function replaces an identifier in all open windows, after verifying that the new identifier does not exist in any window (unless the new identifier matches the original one except for case). This is useful for changing identifiers in large projects without missing them in some files and without getting conflicting identifiers. Of course, all files that belong to the project must be open in order for this to work correctly. OTOH, be sure not to do unwanted replacements in windows opened that don't belong to the project. The search and replacement used here is always case-insensitive.

10.4 The '`Run`' Menu

Besides the items listed here, this menu may contain additional tools and variables defined with a '`/Run`' prefix (see Chapter 11 [Tools], page 63).

10.4.1 The PENG Built-In Tool '`Run/Run`'

Compiles and links a source file, automatically updating the units used when necessary, like '`Compile/Make`' (see Section 10.5.2 [Menu Compile/Make], page 33) does, and then starts the resulting executable. The file to compile is the one in the current editor, or the one given under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33) if it is given.

Currently implemented for Pascal using GPC (see section "GNU Pascal" in *the GNU Pascal manual*), C using GCC (see section "Introduction" in *the GCC manual*), LaTeX (see section "Overview" in *the LaTeX manual*) and Texinfo (see section "Texinfo" in *the Texinfo manual*). For LaTeX and Texinfo, *starting the executable* means opening the document created in '`xdvi`' (if possible) or in PENG's info reader, respectively.

10.4.2 The PENG Built-In Tool Variable '`Run/Arguments`'

Here you can specify command line arguments that are given when running a compiled executable with '`Run/Run`' (see Section 10.4.1 [Menu Run/Run], page 32).

Effective for Pascal and C.

10.5 The '`Compile`' Menu

Besides the items listed here, this menu may contain additional tools and variables defined with a '`/Compile`' prefix (see Chapter 11 [Tools], page 63).

When both quote protection and paragraph mode (see Section 10.7.1.6 [Menu Options/Editor1/Paragraphs], page 44) are on, inserting a line with ⟨ENTER⟩ into a quoted line will also cause unquoted empty lines to be inserted between the parts of the line, and the cursor to be placed onto the empty lines, which is very practical for replying to mail and news.

Like paragraph mode, quote protection does not affect block operations, for the same reason. Quote protection also disables automatic reformatting of quoted paragraphs (e.g., after splitting or merging them) when paragraph mode is on.

10.7.1.10 The Menu Entry '`Options/Editor1/Line numbers`'

If this option is on, text editors will show the line numbers of all lines on the screen at the left side. The width reserved for line numbers can be set under '`Options/Editor2/Line number width`' (see Section 10.7.2.19 [Menu Options/Editor2/Line number width], page 49).

10.7.2 The '`Options/Editor2`' Submenu

The editor options have been split into two submenus, this one and '`Options/Editor1`' (see Section 10.7.1 [Menu Options/Editor1], page 43) (see the rationale given there).

Generally, this menu contains those options that are often only set once after installing PENG and rarely touched later, while '`Options/Editor1`' contains those that are typically changed more frequently. However, the distinction may be a bit arbitrary in some cases.

10.7.2.1 The Menu Entry '`Options/Editor2/Backups`'

PENG can make either simple or multiple (numbered) backups, each in either a *short* form ('`.bak`'; '`.b00`', '`.b01`', ...) or a *long* form ('`~`'; '`.~1~`', '`.~2~`', ...). Under Dos, of course, only the short form is available.

10.7.2.2 The Menu Entry '`Options/Editor2/Backup directory`'

Directory in which to create backup files (see Section 10.7.2.1 [Menu Options/Editor2/Backups], page 45). Defaults to a temp directory, so that backup files don't clutter the data directories. It might be useful to have old files in that directory automatically removed by a cron job.

10.7.2.3 The Menu Entry '`Options/Editor2/Clipboard file`'

If a file name is entered here, the clipboard will be stored in this file, so it survives quitting and restarting PENG. However, since PENG only appends to the clipboard, the file should regularly be cleared, either manually, or with '`Edit/Clear clipboard`' (see Section 10.2.9 [Menu Edit/Clear clipboard], page 28), or by a cron job. If no file name is entered, the clipboard will not be saved in a file, and its contents will be lost when quitting PENG.

tions/Editor1/Quote string], page 44) and the *mark string* (see Section 10.7.1.8 [Menu Options/Editor1/Mark string], page 44) to match the previous line.

10.7.1.6 The Menu Entry '`Options/Editor1/Paragraphs`'

If this option is on, blocks of lines that have the same indentation and quotation level, determined by leading spaces and (possibly multiple) leading occurences of the *quote string* (see Section 10.7.1.7 [Menu Options/Editor1/Quote string], page 44) and the *mark string* (see Section 10.7.1.8 [Menu Options/Editor1/Mark string], page 44) are treated as paragraphs, separated by empty lines.

In particular, the cursor and words are wrapped automatically between the lines when they reach the length given under '`Options/Editor2/Line width`' (see Section 10.7.2.18 [Menu Options/Editor2/Line width], page 48). ⟨ENTER⟩ and ⟨CTRL⟩-⟨N⟩ have twice the usual effect, so they create an empty line and start a new paragraph.

This mode is useful, e.g., for editing mail, news, TeX, HTML, etc. It can be applied to certain files only (see Section 10.7.2.17 [Menu Options/Editor2/Paragraphs in files], page 48).

This mode implies '`Trim trailing spaces`' (see Section 10.7.2.11 [Menu Options/Editor2/Trim trailing spaces], page 47), because the notion of "trailing" is quite pointless when considering lines as parts of paragraphs.

Please note: Block operations and line deletions with ⟨CTRL⟩-⟨Y⟩ are not affected, so they can be used as an escape to override the paragraph structure.

10.7.1.7 The Menu Entry '`Options/Editor1/Quote string`'

Here you can specify a string considered to start each line of a *quotation* in a text (usually '`> `' in mail and news messages). This is relevant for quote protection (see Section 10.7.1.9 [Menu Options/Editor1/Protect quotes], page 44), indentation (see Section 10.7.1.5 [Menu Options/Editor1/Indent], page 43), and can be inserted before each line of a block of text (see Section 5.9 [Text Editor Keys], page 13).

10.7.1.8 The Menu Entry '`Options/Editor1/Mark string`'

Here you can specify a string considered to start each line of some markup in a text (e.g., '`: `' to mark some included foreign text in mail and news messages). This is relevant for indentation (see Section 10.7.1.5 [Menu Options/Editor1/Indent], page 43), and can be inserted before each line of a block of text (see Section 5.9 [Text Editor Keys], page 13).

10.7.1.9 The Menu Entry '`Options/Editor1/Protect quotes`'

If this option is on, quoted lines can only be split and merged, but not edited otherwise without special confirmation. If confirmation is given, it will remove the protection for the current file, until it is closed and reopened, or PENG is restarted.

10.5.1 The PENG Built-In Tool '`Compile/Compile`'

Compiles a source file without trying to link it. The file to compile is the one in the current editor, or the one given under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33) if it is given.

Currently implemented for Pascal using GPC (see section "GNU Pascal" in *the GNU Pascal manual*), C using GCC (see section "Introduction" in *the GCC manual*), LaTeX (see section "Overview" in *the LaTeX manual*) and Texinfo (see section "Texinfo" in *the Texinfo manual*).

10.5.2 The PENG Built-In Tool '`Compile/Make`'

Compiles and links a source file, automatically updating the units used when necessary. The file to compile is the one in the current editor, or the one given under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33) if it is given.

Currently implemented for Pascal using GPC (see section "GNU Pascal" in *the GNU Pascal manual*), C using GCC (see section "Introduction" in *the GCC manual*), LaTeX (see section "Overview" in *the LaTeX manual*) and Texinfo (see section "Texinfo" in *the Texinfo manual*). For LaTeX and Texinfo, it does the same as '`Compile/Compile`' (see Section 10.5.1 [Menu Compile/Compile], page 33).

10.5.3 The PENG Built-In Tool '`Compile/Build`'

Compiles a source file, recompiling all units used whether changed or not. The file to compile is the one in the current editor, or the one given under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33) if it is given.

Currently implemented only for Pascal using GPC (see section "GNU Pascal" in *the GNU Pascal manual*).

10.5.4 The PENG Built-In Tool '`Compile/Set current file as main file`'

Sets the file name of the current editor as the main file, as if you had entered it under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33).

10.5.5 The PENG Built-In Tool '`Compile/Clear main file`'

Clears the main file, as if you had entered an empty string under '`Compile/Main file`' (see Section 10.5.6 [Menu Compile/Main file], page 33).

10.5.6 The PENG Built-In Tool Variable '`Compile/Main file`'

If you enter a file name here, this file will be used rather than the current file for all compile operations such as '`Compile/Compile`' (see Section 10.5.1 [Menu Compile/Compile], page 33),

'`Compile/Make`' (see Section 10.5.2 [Menu Compile/Make], page 33), '`Compile/Build`' (see Section 10.5.3 [Menu Compile/Build], page 33) and '`Run/Run`' (see Section 10.4.1 [Menu Run/Run], page 32).

You can also set or clear this entry with '`Compile/Set current file as main file`' (see Section 10.5.4 [Menu Compile/Set current file as main file], page 33) or '`Compile/Clear main file`' (see Section 10.5.5 [Menu Compile/Clear main file], page 33), respectively.

10.6 The '`Tools`' Menu

Besides the items listed here, this menu may contain additional tools and variables defined with a '`/Tools`' prefix or no prefix at all (see Chapter 11 [Tools], page 63).

10.6.1 The '`Tools/Macros`' Submenus

10.6.1.1 The PENG Built-In Tool '`Tools/Macros/Pascal/New program`'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new Pascal program in the current text editor.

10.6.1.2 The PENG Built-In Tool '`Tools/Macros/Pascal/New program with units`'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new Pascal program with an '`uses`' statement (see section "uses" in *the GNU Pascal manual*) in the current text editor.

10.6.1.3 The PENG Built-In Tool '`Tools/Macros/Pascal/procedure`'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a procedure declaration (see section "procedure" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.4 The PENG Built-In Tool '`Tools/Macros/Pascal/function`'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a function declaration (see section "function" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.5 The PENG Built-In Tool '`Tools/Macros/Pascal/block`'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`begin ... end`' block (see section "begin" in *the GNU Pascal manual*) at the current cursor position.

10.7 The '`Options`' Menu

Besides the items listed here, this menu may contain additional tools and variables defined with an '`/Options`' prefix (see Chapter 11 [Tools], page 63).

10.7.1 The '`Options/Editor1`' Submenu

The editor options have been split into two submenus, this one and '`Options/Editor2`' (see Section 10.7.2 [Menu Options/Editor2], page 45), because they were becoming too many for one menu (though there is no limit on the size of a menu in PENG, considerations about screen size and available shortcut keys indicate the need for separate menus).

Generally, this menu contains those options that are typically changed more frequently, while '`Options/Editor2`' contains those that are often only set once after installing PENG and rarely touched later. However, the distinction may be a bit arbitrary in some cases.

10.7.1.1 The Menu Entry '`Options/Editor1/Auto copy`'

If this option is on, marked text will automatically be copied to the clipboard, like '`Edit/Copy`' does (see Section 10.2.4 [Menu Edit/Copy], page 28). An exception is when it is deleted (with '`Edit/Delete`' (see Section 10.2.6 [Menu Edit/Delete], page 28) or ⟨CTRL⟩-⟨DEL⟩) immediately after marking, because otherwise '`Edit/Delete`' would do the same as '`Edit/Cut`' (see Section 10.2.3 [Menu Edit/Cut], page 27).

10.7.1.2 The Menu Entry '`Options/Editor1/Persistent blocks`'

If this option is off, blocks can be marked with ⟨SHIFT⟩ and are cleared as soon as the cursor is moved while ⟨SHIFT⟩ is not pressed. If it is on, blocks can be marked with ⟨SHIFT⟩ or with ⟨CTRL⟩-⟨K⟩⟨B⟩/⟨CTRL⟩-⟨K⟩⟨K⟩ (see Section 5.9 [Text Editor Keys], page 13), and the cursor can be moved independently of the marked block.

10.7.1.3 The Menu Entry '`Options/Editor1/Delete blocks`'

If this option is on, ⟨DEL⟩ and ⟨BACKSPACE⟩ will delete a block if one is marked, and have their normal function otherwise.

10.7.1.4 The Menu Entry '`Options/Editor1/Volatile blocks`'

If this option is on, marked blocks will be deleted before insertions, including ⟨TAB⟩, ⟨CTRL⟩-⟨V⟩, ⟨CTRL⟩-⟨N⟩, and ⟨ENTER⟩ when used for insertion.

10.7.1.5 The Menu Entry '`Options/Editor1/Indent`'

If this option is on, new lines inserted into a text editor will be indented with spaces, tabs and (possibly multiple) occurences of the *quote string* (see Section 10.7.1.7 [Menu Op-

10.6.10 The PENG Built-In Tool 'Tools/ChangeLog entry'

Opens the ChangeLog file specified under 'Tools/Settings/ChangeLog file name' (see Section 10.6.13.3 [Menu Tools/Settings/ChangeLog file name], page 42) and inserts a new entry at the top, using the current date and your name and email address specified under 'Tools/Settings/Name' (see Section 10.6.13.1 [Menu Tools/Settings/Name], page 42) and 'Tools/Settings/Mail address' (see Section 10.6.13.2 [Menu Tools/Settings/Mail address], page 42).

10.6.11 The PENG Built-In Tool 'Tools/Calendar'

Shows a calendar of the current month in a read-only window, using the 'gcal' program (see section "Gcal" in *the Gcal manual*). I don't know why, but it seems that an IDE *must* have a calendar function... ;-)

10.6.12 The PENG Built-In Tool Variable 'Tools/Files to grep'

Here you can specify a list of files that the 'Tools/Grep' tool (see Section 10.6.2 [Menu Tools/Grep], page 40) will search by default. For any particular 'grep' invocation, you can override the default.

The string entered here is given to the shell, and therefore may contain any wildcard characters your shell understands. The default will search all files with some known *source* extensions in the current directory.

10.6.13 The 'Tools/Settings' Submenu

10.6.13.1 The PENG Built-In Tool Variable 'Tools/Settings/Name'

Your real name which you can enter here will be used by the 'Tools/ChangeLog entry' tool (see Section 10.6.10 [Menu Tools/ChangeLog entry], page 42), but, of course, you can define other tools using this variable if they need the real name and you want to enter it in only one place.

10.6.13.2 The PENG Built-In Tool Variable 'Tools/Settings/Mail address'

Your email address which you can enter here will be used by the 'Tools/ChangeLog entry' tool (see Section 10.6.10 [Menu Tools/ChangeLog entry], page 42), but, of course, you can define other tools using this variable if they need the email address and you want to enter it in only one place.

10.6.13.3 The PENG Built-In Tool Variable 'Tools/Settings/ChangeLog file name'

Here you can specifiy the name of the ChangeLog file in which you want 'Tools/ChangeLog entry' (see Section 10.6.10 [Menu Tools/ChangeLog entry], page 42) to insert your entries.

10.6.1.6 The PENG Built-In Tool 'Tools/Macros/Pascal/if then'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an 'if ... then' statement (see section "if" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.7 The PENG Built-In Tool 'Tools/Macros/Pascal/if then else'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an 'if ... then ... else' statement (see section "else" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.8 The PENG Built-In Tool 'Tools/Macros/Pascal/case'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a 'case' statement (see section "case" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.9 The PENG Built-In Tool 'Tools/Macros/Pascal/while loop'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a 'while' loop (see section "while" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.10 The PENG Built-In Tool 'Tools/Macros/Pascal/repeat loop'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a 'repeat' loop (see section "repeat" in *the GNU Pascal manual*) at the current cursor position.

10.6.1.11 The PENG Built-In Tool 'Tools/Macros/Pascal/list type'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a list type declaration at the current cursor position, after prompting for its name. The name will automatically be prefixed with a 'P' for the pointer type and 'T' for the actual record.

10.6.1.12 The PENG Built-In Tool 'Tools/Macros/Pascal/object type'

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an object type declaration (see section "object" in *the GNU Pascal manual*) at the current cursor position, after prompting for its name. The name will automatically be prefixed with a 'P' for the pointer type and 'T' for the actual object.

10.6.1.13 The PENG Built-In Tool ‘Tools/Macros/Pascal/walk through a list’

This Pascal syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts code to walk a pointer through a single linked list at the current cursor position, after prompting for the name of the variable to be used.

10.6.1.14 The PENG Built-In Tool ‘Tools/Macros/C/New program’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new C program in the current text editor.

10.6.1.15 The PENG Built-In Tool ‘Tools/Macros/C/New program with includes’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new C program with some ‘#include’ directives in the current text editor.

10.6.1.16 The PENG Built-In Tool ‘Tools/Macros/C/void function’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘void’ function declaration at the current cursor position.

10.6.1.17 The PENG Built-In Tool ‘Tools/Macros/C/function’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a function declaration at the current cursor position.

10.6.1.18 The PENG Built-In Tool ‘Tools/Macros/C/block’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘{ ... }’ block at the current cursor position.

10.6.1.19 The PENG Built-In Tool ‘Tools/Macros/C/if’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an ‘if’ statement at the current cursor position.

10.6.1.20 The PENG Built-In Tool ‘Tools/Macros/C/if else’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an ‘if ... else’ statement at the current cursor position.

10.6.3 The PENG Built-In Tool ‘Tools/Calculator’

Lets you enter a mathematical expression using real or complex numbers, and evaluates it. (What a surprise! ;–) For the operations available, see Section 11.6.27 [Tool MATH], page 73.

10.6.4 The PENG Built-In Tool ‘Tools/Evaluate’

Lets you enter an expression to be evaluated. Currently, this is the same as ‘Tools/Calculator’ (see Section 10.6.3 [Menu Tools/Calculator], page 41), but when PENG will support a debugger, it will use the debugger's functionality to evaluate expressions in the source language, using variables from the running program etc. ...

10.6.5 The PENG Built-In Tool ‘Tools/Man page’

Opens a man page in a read-only window, so you can easily copy and paste from it. Of course, you can also enter the man page section, e.g., ‘1 man’ to get the manual of the ‘man’ program in section 1.

10.6.6 The PENG Built-In Tool ‘Tools/Create orig’

Copies the current file to a file name obtained by appending ‘.orig’ to the current file name. Useful if you want to make changes to a file and create a diff later (see section “Overview” in *the diff manual*). Uses the ‘-i’ option to ‘cp’, so it won't overwrite an existing ‘.orig’ file without confirmation.

10.6.7 The PENG Built-In Tool ‘Tools/Insert uuencoded’

Lets you select a file, and inserts it into the current editor after uuencoding it. Removes the original file after asking for confirmation.

10.6.8 The PENG Built-In Tool ‘Tools/Insert gzipped and uuencoded’

Lets you select a file, and inserts it into the current editor after uuencoding and gzipping it. Removes the original file after asking for confirmation.

10.6.9 The PENG Built-In Tool ‘Tools/Rename’

Lets you select a file, and rename or move it, while you can edit the original file name to get the new one (especially useful for small changes in long file names).

Of course, PENG is not going to be a complete file manager, but some small jobs can easily be done using the available PENG tool functions (see Section 11.6 [Tool Functions], page 67)...

10.6.1.45 The PENG Built-In Tool ‘`Tools/Macros/texi/email`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an ‘`@email{}`’ command (see section “email” in *the Texinfo manual*) at the current cursor position.

10.6.1.46 The PENG Built-In Tool ‘`Tools/Macros/texi/uref`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an ‘`@uref{}`’ command (see section “uref” in *the Texinfo manual*) at the current cursor position.

10.6.1.47 The PENG Built-In Tool ‘`Tools/Macros/texi/kbd`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`@kbd{@key{}}`’ command (see section “kbd” in *the Texinfo manual*) at the current cursor position.

10.6.1.48 The PENG Built-In Tool ‘`Tools/Macros/texi/key`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`@key{}`’ command (see section “key” in *the Texinfo manual*) at the current cursor position.

10.6.1.49 The PENG Built-In Tool ‘`Tools/Macros/texi/pxref`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an ‘`@pxref{}`’ command (see section “pxref” in *the Texinfo manual*) at the current cursor position.

10.6.1.50 The PENG Built-In Tool ‘`Tools/Macros/texi/ref`’

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`@ref{}`’ command (see section “ref” in *the Texinfo manual*) at the current cursor position.

10.6.2 The PENG Built-In Tool ‘`Tools/Grep`’

Invokes ‘`grep`’ to find something (by default, the current word in the current editor) in a list of files (by default, all *source* files in the current directory), and shows the results in PENG’s message window, so you can easily view or edit the places found in a text editor.

Under ‘`Tools/Files to grep`’ (see Section 10.6.12 [Menu Tools/Files to grep], page 42), you can change the default list of files, e.g., to provide a list of directories where you keep your source files, so they will always be searched.

10.6.1.21 The PENG Built-In Tool ‘`Tools/Macros/C/switch`’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`switch`’ statement at the current cursor position.

10.6.1.22 The PENG Built-In Tool ‘`Tools/Macros/C/while loop`’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`while`’ loop at the current cursor position.

10.6.1.23 The PENG Built-In Tool ‘`Tools/Macros/C/do while loop`’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`do` ... `while`’ loop at the current cursor position.

10.6.1.24 The PENG Built-In Tool ‘`Tools/Macros/C/list type`’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a list type declaration at the current cursor position, after prompting for its name.

10.6.1.25 The PENG Built-In Tool ‘`Tools/Macros/C/walk through a list`’

This C syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts code to walk a pointer through a single linked list at the current cursor position, after prompting for the name of the variable to be used.

10.6.1.26 The PENG Built-In Tool ‘`Tools/Macros/LaTeX/New text`’

This LaTeX syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new LaTeX document in the current text editor.

10.6.1.27 The PENG Built-In Tool ‘`Tools/Macros/LaTeX/block`’

This LaTeX syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a ‘`\begin{`*foo*`}` ... `\end{`*foo*`}`’ environment (see section “Environments” in *the LaTeX manual*) at the current cursor position, after prompting for its name (*foo*).

10.6.1.28 The PENG Built-In Tool ‘`Tools/Macros/LaTeX/enumerate`’

This LaTeX syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a skeleton for an enumeration (see section “enumerate” in *the LaTeX manual*) at the current cursor position.

10.6.1.29 The PENG Built-In Tool '`Tools/Macros/texi/New text`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) creates the skeleton for a new Texinfo file in the current text editor.

10.6.1.30 The PENG Built-In Tool '`Tools/Macros/texi/chapter`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a node for a new chapter (see section "chapter" in *the Texinfo manual*) at the current cursor position.

10.6.1.31 The PENG Built-In Tool '`Tools/Macros/texi/section`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a node for a new section (see section "section" in *the Texinfo manual*) at the current cursor position.

10.6.1.32 The PENG Built-In Tool '`Tools/Macros/texi/subsection`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a node for a new subsection (see section "subsection" in *the Texinfo manual*) at the current cursor position.

10.6.1.33 The PENG Built-In Tool '`Tools/Macros/texi/subsubsection`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a node for a new subsubsection (see section "subsubsection" in *the Texinfo manual*) at the current cursor position.

10.6.1.34 The PENG Built-In Tool '`Tools/Macros/texi/menu`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a menu (see section "Menus" in *the Texinfo manual*) at the current cursor position.

10.6.1.35 The PENG Built-In Tool '`Tools/Macros/texi/itemize`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an '`@itemize`' command (see section "itemize" in *the Texinfo manual*) at the current cursor position.

10.6.1.36 The PENG Built-In Tool '`Tools/Macros/texi/table`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@table`' (see section "Two-column Tables" in *the Texinfo manual*) command at the current cursor position.

10.6.1.37 The PENG Built-In Tool '`Tools/Macros/texi/multitabe`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@multitable`' command (see section "Multi-column Tables" in *the Texinfo manual*) at the current cursor position.

10.6.1.38 The PENG Built-In Tool '`Tools/Macros/texi/emph`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts an '`@emph{}`' command (see section "emph & strong" in *the Texinfo manual*) at the current cursor position.

10.6.1.39 The PENG Built-In Tool '`Tools/Macros/texi/strong`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@strong{}`' command (see section "emph & strong" in *the Texinfo manual*) at the current cursor position.

10.6.1.40 The PENG Built-In Tool '`Tools/Macros/texi/samp`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@samp{}`' command (see section "samp" in *the Texinfo manual*) at the current cursor position.

10.6.1.41 The PENG Built-In Tool '`Tools/Macros/texi/code`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@code{}`' command (see section "code" in *the Texinfo manual*) at the current cursor position.

10.6.1.42 The PENG Built-In Tool '`Tools/Macros/texi/var`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@var{}`' command (see section "var" in *the Texinfo manual*) at the current cursor position.

10.6.1.43 The PENG Built-In Tool '`Tools/Macros/texi/dfn`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@dfn{}`' command (see section "dfn" in *the Texinfo manual*) at the current cursor position.

10.6.1.44 The PENG Built-In Tool '`Tools/Macros/texi/file`'

This Texinfo syntax macro (see Section 11.6.41.1 [Syntax Macros], page 76) inserts a '`@file{}`' command (see section "file" in *the Texinfo manual*) at the current cursor position.