Stand-alone GNU Info

for version 7.1.1, 7 September 2024

Brian J. Fox
and Texinfo maintainers

This manual is for Stand-alone GNU Info (version 7.1.1, 7 September 2024), a program for viewing documents in Info format (usually created from Texinfo source files).

Table of Contents

1 Stand-alone Info

The *Info* program described here is a stand-alone program, part of the Texinfo distribution, which is used to view Info files on a text terminal. *Info files* are typically the result of processing Texinfo files with the program `texi2any` (also in the Texinfo distribution).

Texinfo itself (see *Texinfo*) is a documentation system that uses a single source file to produce both on-line information and printed output. You can typeset and print the files that you read in Info.

GNU Emacs also provides an Info reader (just type `M-x info` in Emacs). Emacs Info and stand-alone Info have nearly identical user interfaces, although customization and other details are different (this manual explains the stand-alone Info reader). The Emacs Info reader supports the X Window System and other such bitmapped interfaces, not just plain ASCII, so if you want a prettier display for Info files, you should try it. You can use Emacs Info without using Emacs for anything else. (Type `C-x C-c` to exit; this also works in the stand-alone Info reader.) See *Info* for a tutorial and more background information about the Info system, as well as information about the Info reader that is part of GNU Emacs,

Please report bugs in this stand-alone Info program to `bug-texinfo@gnu.org`. Bugs in the Emacs Info reader should be sent to `bug-gnu-emacs@gnu.org`.

2 Selecting a Node

The most basic node commands are '`n`', '`p`', and '`u`'. These move you around the node structure of the file:

n (`next-node`)
: Select the 'Next' node.

p (`prev-node`)
: Select the 'Prev' node.

u (`up-node`)
: Select the 'Up' node.

The top line of each Info node contains *pointers* which describe where the next, previous, and up nodes are. The nodes in an Info file are conventionally arranged in a hierarchical structure; for example, the 'Next' pointer in a section of a chapter takes you to the next section in the same chapter, while the 'Up' pointer takes you to the higher-level node for the chapter as a whole.

You can select a node that you have already viewed by using the '`l`' command—this stands for "last":

l (`history-node`)
: Move backwards through the history of visited nodes for this window. The current node is discarded from the history.

  This is useful when you follow a reference to another node to read about a related issue, and would like then to resume reading at the same place where you started.

Two additional commands, '`t`' and '`d`', select special nodes:

t (`top-node`)
: Select the node '`Top`' in the current Info file.

d (`dir-node`)
: Select the directory node (i.e., the node '`(dir)`'). This contains a menu referencing all the available Info files that are installed on your system.

In the command descriptions in this manual, the *M-x* command names are displayed in parentheses. *M-x* is itself a command (`execute-command`) letting you run a command by name. For example, you could select the next node with *M-x next-node*.

C-x means press the `Ctrl` key and the key *x*. *M-x* means press the `Meta` key and the key *x*. (`Meta` is usually labeled as `Alt`). `SPC` is the space bar. The other keys are usually called by the names printed on them. This notation to describe keystrokes is the same as that used within the Emacs manual (see Section "User Input" in *The GNU Emacs Manual*).

Many Info commands can take a *numeric argument*; see Chapter 11 [Miscellaneous Commands], page 18, to find out how to supply one.

3 Further Commands for Selecting a Node

This chapter describes some other commands which select a different node.

`<` (`first-node`)
: Selects the first node which appears in this file. This node is most often '`Top`', but it does not have to be. With a numeric argument *N*, select the *N*th node (the first node is node 1). An argument of zero is the same as the argument of 1.

`>` (`last-node`)
: Select the last node which appears in this file. With a numeric argument *N*, select the *N*th node (the first node is node 1). An argument of zero is the same as no argument, i.e., it selects the last node.

`]` (`global-next-node`)
: Move forward through the node structure. If the node that you are currently viewing has a menu, select the first menu item. Otherwise, if this node has a '`Next`' pointer, follow it. If there is no menu and no '`Next`' pointer, then follow '`Up`' pointers until there is a '`Next`' pointer, and then follow it.

`[` (`global-prev-node`)
: Move backward through the node structure. If the node that you are currently viewing has a '`Prev`' pointer, that node is selected. Otherwise, if the node has an '`Up`' pointer, that node is selected, and if it has a menu, the last item in the menu is selected.

  You can get the same behavior as `global-next-node` and `global-prev-node` while simply scrolling through the file with `SPC` and `DEL` (see [scroll-behavior], page 29).

`g` (`goto-node`)
: Read the name of a node and select it. If the desired node resides in some other file, you must type the node as it appears in that Info file, and include the name of the other file. For example,

  ```
  g(emacs)Buffers
  ```

  finds the node '`Buffers`' in the Info file `emacs`.

  While reading the node name, completion (see Section 9.3 [The Echo Area], page 14) is only done for the nodes which reside in one of the Info files that were loaded in the current Info session.

`O` (`goto-invocation`)
: Read the name of a program and look for a node in the current Info file which describes the invocation and the command-line options for that program. The default program name is derived from the name of the current Info file. This command does the same as the '`--show-options`' command-line option (see [–show-options], page 25), but it also allows to specify the program name; this is important for those manuals which describe several programs.

  If you need to find the Invocation node of a program that is documented in another Info file, you need to visit that file before invoking '`O`'. For example, if

you are reading the Emacs manual and want to see the command-line options of the `texi2any` program, type *g (texinfo)* RET and then *O texi2any* RET. If you don't know what Info file documents the command, or if invoking '`O`' doesn't display the right node, go to the '`(dir)`' node (using the '`d`' command) and invoke '`O`' from there.

G (`menu-sequence`)
: Read a sequence of menu entries and follow it. Info prompts for a sequence of menu items separated by commas. (Since commas are not allowed in a node name, they are a natural choice for a delimiter in a list of menu items.) Info then looks up the first item in the menu of the node '`(dir)`' (if the '`(dir)`' node cannot be found, Info uses '`Top`'). If such an entry is found, Info goes to the node it points to and looks up the second item in the menu of that node, etc. In other words, you can specify a complete path which descends through the menu hierarchy of a particular Info file starting at the '`(dir)`' node. This has the same effect as if you typed the menu item sequence on Info's command line, see [Info command-line arguments processing], page 21. For example,

  G Texinfo,Overview,Reporting Bugs RET

  displays the node '`Reporting Bugs`' in the Texinfo manual. (You don't actually need to type the menu items in their full length, or in their exact letter-case. However, if you do type the menu items exactly, Info will find it faster.)

  If any of the menu items you type are not found, Info stops at the last entry it did find and reports an error.

C-x C-f (`view-file`)
: Read the name of a file and selects the entire file. The command

  C-x C-f *filename*

  is equivalent to typing

  g(*filename*)*

C-x C-b (`list-visited-nodes`)
: Make a window containing a menu of all of the currently visited nodes. This window becomes the selected window, and you may use the standard Info commands within it.

C-x b (`select-visited-node`)
: Select a node which has been previously visited in a visible window. This is similar to '`C-x C-b`' followed by '`m`', but no window is created.

M-x `man`
: Read the name of a man page to load and display. This uses the `man` command on your system to retrieve the contents of the requested man page. See also [–raw-escapes], page 24.

4 Moving the Cursor

GNU Info has several commands which allow you to move the cursor about the screen.

With a numeric argument, the motion commands are simply executed that many times; for example, a numeric argument of 4 given to `next-line` causes the cursor to move down 4 lines. With a negative numeric argument, the motion is reversed; an argument of −4 given to the `next-line` command would cause the cursor to move *up* 4 lines.

C-n (`next-line`)
`DOWN` (an arrow key)
: Move the cursor down to the next line.

C-p (`prev-line`)
`UP` (an arrow key)
: Move the cursor up to the previous line.

C-a (`beginning-of-line`)
: Move the cursor to the start of the current line.

C-e (`end-of-line`)
: Move the cursor to the end of the current line.

C-f (`forward-char`)
`RIGHT` (an arrow key)
: Move the cursor forward a character.

C-b (`backward-char`)
`LEFT` (an arrow key)
: Move the cursor backward a character.

M-f (`forward-word`)
: Move the cursor forward a word.

M-b (`backward-word`)
: Move the cursor backward a word.

M-< (`beginning-of-node`)
b
: Move the cursor to the start of the current node.

M-> (`end-of-node`)
e
: Move the cursor to the end of the current node.

M-r (`move-to-window-line`)
: Move the cursor to a specific line of the window. Without a numeric argument, `M-r` moves the cursor to the start of the line in the center of the window. With a numeric argument of *n*, `M-r` moves the cursor to the start of the *n*th line in the window.

5 Moving Text Within a Window

Sometimes you are looking at a screenful of text, and only part of the current paragraph you are reading is visible on the screen. The commands detailed in this section are used to shift which part of the current node is visible on the screen.

SPC (scroll-forward)

NEXT Shift the text in this window up. That is, show more of the node which is currently below the bottom of the window. With a numeric argument, show that many more lines at the bottom of the window; a numeric argument of 4 would shift all of the text in the window up 4 lines (discarding the top 4 lines), and show you four new lines at the bottom of the window. Without a numeric argument, SPC takes the bottom two lines of the window and places them at the top of the window, redisplaying almost a completely new screenful of lines. If you are at the end of a node, SPC takes you to the "next" node, so that you can read an entire manual from start to finish by repeating SPC.

The NEXT key is known as the PageDown key on some keyboards.

C-v (scroll-forward-page-only)

Shift the text in this window up. This is identical to the SPC operation above, except that it never scrolls beyond the end of the current node.

M-x scroll-forward-page-only-set-window

Scroll forward, like with *C-v*, but if a numeric argument is specified, it becomes the default scroll size for subsequent scroll-forward and scroll-backward commands and their ilk.

DEL (scroll-backward)

PREVIOUS Shift the text in this window down. The inverse of scroll-forward. If you are at the start of a node, DEL takes you to the "previous" node, so that you can read an entire manual from finish to start by repeating DEL. The default scroll size can be changed by invoking the (scroll-backward-page-only-set-window) command with a numeric argument.

If your keyboard lacks the DEL key, look for a key called BS, or 'Backspace', sometimes designated with an arrow which points to the left, which should perform the same function.

The PREVIOUS key is the PageUp key on many keyboards. Emacs refers to it by the name PRIOR.

M-v (scroll-backward-page-only)

Shift the text in this window down. The inverse of scroll-forward-page-only. Does not scroll beyond the start of the current node. The default scroll size can be changed by invoking the scroll-backward-page-only-set-window command with a numeric argument.

M-x scroll-backward-page-only-set-window

Scroll backward, like with *M-v*, but if a numeric argument is specified, it becomes the default scroll size for subsequent scroll-forward and scroll-backward commands.

M-x `down-line`
: Scroll forward by one line. With a numeric argument, scroll forward that many lines.

M-x `up-line`
: Scroll backward one line. With a numeric argument, scroll backward that many lines.

M-x `scroll-half-screen-down`
: Scroll forward by half of the screen size. With a numeric argument, scroll that many lines. If an argument is specified, it becomes the new default number of lines to scroll for subsequent `scroll-half-screen-down` and `scroll-half-screen-up` commands.

M-x `scroll-half-screen-up`
: Scroll back by half of the screen size. With a numeric argument, scroll that many lines. If an argument is specified, it becomes the new default number of lines to scroll for subsequent `scroll-half-screen-down` and `scroll-half-screen-up` commands.

The `scroll-forward` and `scroll-backward` commands can also move forward and backward through the node structure of the file. If you press `SPC` while viewing the end of a node, or `DEL` while viewing the beginning of a node, what happens is controlled by the variable `scroll-behavior` (see [scroll-behavior], page 29).

The `scroll-forward-page-only` and `scroll-backward-page-only` commands never scroll beyond the current node.

6 Searching an Info File

GNU Info allows you to search for a sequence of characters throughout an entire Info file. Here are the commands to do this:

`s` (`search`)

`/`
: Read a string in the echo area and search for it, either as a regular expression (by default) or a literal string. If the string includes upper-case characters, the Info file is searched case-sensitively; otherwise Info ignores the letter case. With a numeric argument of *N*, search for *N*th occurrence of the string. Negative arguments search backwards.

`?` (`search-backward`)
: Read a string in the echo area and search backward through the Info file for that string. If the string includes upper-case characters, the Info file is searched case-sensitively; otherwise Info ignores the letter case. With a numeric argument of *N*, search for *N*th occurrence of the string. Negative arguments search forward.

`C-x n` (`search-next`)

`}`
: Search forwards for the string used for the last search command. Case sensitivity and use of regular expressions are kept the same. With a numeric argument of *n*, search for *n*th next occurrence.

  By default, the search starts at the position immediately following the cursor. However, if the variable `search-skip-screen` (see Chapter 13 [`search-skip-screen`], page 27) is set, it starts at the beginning of the next page, thereby skipping all visibly displayed lines.

`C-x N` (`search-previous`)

`{`
: Just like `search-next`, but in reverse. You can use `search-next` and `search-previous` together to move forward and backward through matches. `search-previous` usually goes to the place in the file that was displayed before an immediately preceding `search-next`, and vice versa.[1]

`R` (`toggle-regexp`)
: Toggle between using regular expressions and literal strings for searching. Info uses so-called 'extended' regular expression syntax (see Section "Regular Expressions" in *GNU Grep*).

`S` (`search-case-sensitively`)
: Read a string in the echo area and search for it case-sensitively, even if the string includes only lower-case letters. With a numeric argument of *N*, search for *N*th occurrence of the string. Negative arguments search backwards.

`C-s` (`isearch-forward`)
: Interactively search forward through the Info file for a string as you type it. If the string includes upper-case characters, the search is case-sensitive; otherwise Info ignores the letter case.

[1] This sometimes doesn't happen when `search-skip-screen` is `On`, and the search goes across nodes.

C-r (`isearch-backward`)
: Interactively search backward through the Info file for a string as you type it. If the string includes upper-case characters, the search is case-sensitive; otherwise Info ignores the letter case.

M-/ (`tree-search`)
: Recursively search this node and any subnodes listed in menus for a string.

M-} (`tree-search-next`)
M-{ (`tree-search-previous`)
: Go forwards and backwards through the matches for an active tree search.

The most basic searching command is '`s`' or '`/`' (`search`). The '`s`' command prompts you for a string in the echo area, and then searches the remainder of the Info file for an occurrence of that string. If the string is found, the node containing it is selected, and the cursor is left positioned at the start of the found string. Subsequent '`s`' commands show you the default search string; pressing `RET` instead of typing a new string will use the default search string.

Incremental searching is similar to basic searching, but the string is looked up while you are typing it, instead of waiting until the entire search string has been specified.

The tree search can be used from the `dir` node to search through all Info files installed on the system. It can also be used to search through a particular chapter of a manual when you are not interested in matches in other chapters.

If the `highlight-searches` variable is set, matches from search commands will be highlighted. See Chapter 13 [`highlight-searches`], page 27. Use the *M-x* `clear-search` command to clear any search highlights.

Both incremental and non-incremental search by default ignore the case of letters when comparing the Info file text with the search string. However, an uppercase letter in the search string makes the search case-sensitive. You can force a case-sensitive non-incremental search, even for a string that includes only lower-case letters, by using the '`S`' command (`search-case-sensitively`). The '`n`' and '`N`' commands operate case-sensitively if the last search command was '`S`'.

Normally, the search pattern should not be shorter than some predefined limit. By default, this limit is set to 1 character. See [min-search-length], page 29, for more information on this.

7 Index Commands

GNU Info has commands to search through the indices of an Info file, which helps you find areas within an Info file which discuss a particular topic.

`i` (`index-search`)
: Look up a string in the indices for this Info file, and select a node to which the found index entry points.

`I` (`virtual-index`)
: Look up a string in the indices for this Info file, and show all the matches in a new virtual node, synthesized on the fly.

`,` (`next-index-match`)
: Move to the node containing the next matching index item from the last '`i`' command.

`M-x index-apropos`
: Grovel the indices of all the known Info files on your system for a string, and build a menu of the possible matches.

The most efficient means of finding something quickly in a manual is the '`i`' command (`index-search`). This command prompts for a string, and then looks for that string in all the indices of the current Info manual. If it finds a matching index entry, it displays the node to which that entry refers and prints the full text of the entry in the echo area. You can press '`,`' (`next-index-match`) to find more matches. A good Info manual has all of its important concepts indexed, so the '`i`' command lets you use a manual as a reference.

If you don't know what manual documents something, try the *`M-x index-apropos`* command. It prompts for a string and then looks up that string in all the indices of all the Info documents installed on your system. It can also be invoked from the command line; see [–apropos], page 22.

8 Selecting Cross-references

We have already discussed the '`Next`', '`Prev`', and '`Up`' pointers which appear at the top of a node, referring you to a different node.

In addition, a node may contain a *menu*, as well as *cross-references* (*xrefs* for short) interspersed through the text of the node. Cross-references may possibly refer to a node in another Info file.

8.1 Parts of an Xref

Here is a sample menu entry showing the parts of a cross-reference:

```
* Foo Label: Foo Target.        More information about Foo.
```

The reference has two parts: the first part is called the *label*; it is the name that you can use to refer to the cross-reference, and the second is the *target*; it is the full name of the node that the cross-reference points to. The target is separated from the label by a single colon '`:`'; first the label appears, and then the target.

The '`.`' ends the name of the target. The '`.`' is not part of the target; it serves only to let Info know where the target name ends.

A shorthand way of specifying references allows two adjacent colons to stand for a target name which is the same as the label name:

```
* Foo Commands::                Commands pertaining to Foo.
```

In the above example, the name of the target is the same as the name of the label, in this case `Foo Commands`.

You will normally see two types of cross-reference while viewing nodes: *menu* references, and *note* references. Menu references appear within a node's menu; they begin with a '`*`' at the beginning of a line, and continue with a label, a target, and a comment which describes what the contents of the node pointed to contains.

Note references appear within the body of the node text; they begin with `*Note`, and continue with a label and a target.

Like '`Next`', '`Prev`', and '`Up`' pointers, cross-references can point to any valid node. They are used to refer you to a place where more detailed information can be found on a particular subject. Here is a cross-reference which points to a node within the Texinfo documentation: See Section "Cross-refrences" in *the Texinfo Manual*, for more information on creating your own Texinfo cross-references.

8.2 Selecting Xrefs

The following table lists the Info commands which operate on menu items.

1 (`menu-digit`)
2 ... *9*
M-1, vi-like operation
M-2 ... *M-9*, vi-like operation

Within an Info window, pressing a single digit, (such as '`1`'), selects that menu item, and places its node in the current window. For convenience, there is one exception; pressing '`0`' selects the *last* item in the node's menu. When

'`--vi-keys`' is in effect, digits set the numeric argument, so these commands are remapped to their '`M-`' varieties. For example, to select the last menu item, press *M-0*.

0 (`last-menu-item`)
M-0, vi-like operation
: Select the last item in the current node's menu.

m (`menu-item`)
: Reads the name of a menu item in the echo area and selects its node. Completion is available while reading the menu label. See Section 9.3 [The Echo Area], page 14.

M-x `find-menu`
: Move the cursor to the start of this node's menu.

This table lists the Info commands which operate on cross-references.

f (`xref-item`)
r
: Reads the name of a note cross-reference in the echo area and selects its node. Completion is available while reading the cross-reference label. See Section 9.3 [The Echo Area], page 14.

Finally, the next few commands operate on menu or note references alike:

`TAB` (`move-to-next-xref`)
: Move the cursor to the start of the next nearest menu item or note reference in this node. You can then use `RET` (`select-reference-this-line`) to select the menu or note reference.

M-`TAB` (`move-to-prev-xref`)
BackTab
: Move the cursor the start of the nearest previous menu item or note reference in this node.

  The *BackTab* key can be produced on some terminals with *Shift-*`TAB`.

`RET` (`select-reference-this-line`)
: Select the menu item or note reference appearing on this line.

9 Manipulating Multiple Windows

A *window* is a place to show the text of a node. Windows have a view area where the text of the node is displayed, and an associated *mode line*, which briefly describes the node being viewed.

GNU Info supports multiple windows appearing in a single screen; each window is separated from the next by its mode line. At any time, there is only one *active* window, that is, the window in which the cursor appears. There are commands available for creating windows, changing the size of windows, selecting which window is active, and for deleting windows.

9.1 The Mode Line

A *mode line* is a line of inverse video which appears at the bottom of an Info window. It describes the contents of the window just above it; this information includes the name of the file and node appearing in that window, the number of screen lines it takes to display the node, and the percentage of text that is above the top of the window.

Here is a sample mode line for a window containing a file named `dir`, showing the node '`Top`'.

```
-----Info: (dir)Top, 40 lines --Top-------------------------------------
            ^^   ^   ^^^         ^^

          (file)Node #lines    where
```

Truncation of long lines (as opposed to wrapping them to the next display line, see Chapter 5 [Scrolling Commands], page 6) is indicated by a '`$`' at the left edge of the mode line:

```
--$--Info: (texinfo)Top, 480 lines --Top-------------------------------
```

When Info makes a node internally, such that there is no corresponding info file on disk, the name of the node is surrounded by asterisks ('`*`'). The name itself tells you what the contents of the window are; the sample mode line below shows an internally constructed node showing possible completions:

```
-----Info: *Completions*, 7 lines --All--------------------------------
```

9.2 Window Commands

It can be convenient to view more than one node at a time. To allow this, Info can display more than one *window*. Each window has its own mode line (see Section 9.1 [The Mode Line], page 13) and history of nodes viewed in that window (see Chapter 2 [`history-node`], page 2).

C-x o (`next-window`)
: Select the next window on the screen. Note that the echo area can only be selected if it is already in use, and you have left it temporarily. Normally, '`C-x o`' simply moves the cursor into the next window on the screen, or if you are already within the last window, into the first window on the screen. Given a numeric argument, '`C-x o`' moves over that many windows. A negative argument causes '`C-x o`' to select the previous window on the screen.

M-x `prev-window`
: Select the previous window on the screen. This is identical to '`C-x o`' with a negative argument.

C-x 2 (`split-window`)
: Split the current window into two windows, both showing the same node. Each window is one half the size of the original window, and the cursor remains in the original window. The variable `automatic-tiling` can cause all of the windows on the screen to be resized for you automatically (see Chapter 13 [`automatic-tiling`], page 27).

C-x 0 (`delete-window`)
: Delete the current window from the screen. If you have made too many windows and your screen appears cluttered, this is the way to get rid of some of them.

C-x 1 (`keep-one-window`)
: Delete all of the windows excepting the current one.

`ESC` *C-v* (`scroll-other-window`)
: Scroll the other window, in the same fashion that '`C-v`' might scroll the current window. Given a negative argument, scroll the "other" window backward.

C-x ^ (`grow-window`)
: Grow (or shrink) the current window. Given a numeric argument, grow the current window that many lines; with a negative numeric argument, shrink the window instead.

C-x t (`tile-windows`)
: Divide the available screen space among all of the visible windows. Each window is given an equal portion of the screen in which to display its contents. The variable `automatic-tiling` can cause `tile-windows` to be called when a window is created or deleted. See Chapter 13 [`automatic-tiling`], page 27.

9.3 The Echo Area

The *echo area* is a one line window which appears at the bottom of the screen. It is used to display informative or error messages, and to read lines of input from you when that is necessary. Almost all of the commands available in the echo area are identical to their Emacs counterparts, so please refer to that documentation for greater depth of discussion on the concepts of editing a line of text. The following table briefly lists the commands that are available while input is being read in the echo area:

C-f (`echo-area-forward`)
`RIGHT` (an arrow key)
: Move forward a character.

C-b (`echo-area-backward`)
`LEFT` (an arrow key)
: Move backward a character.

C-a (`echo-area-beg-of-line`)
: Move to the start of the input line.

`C-e` (`echo-area-end-of-line`)
: Move to the end of the input line.

`M-f` (`echo-area-forward-word`)
: Move forward a word.

`M-b` (`echo-area-backward-word`)
: Move backward a word.

`C-d` (`echo-area-delete`)
: Delete the character under the cursor.

`DEL` (`echo-area-rubout`)
: Delete the character behind the cursor.

  On some keyboards, this key is designated `BS`, for '`Backspace`'. Those keyboards will usually bind `DEL` in the echo area to `echo-area-delete`.

`C-g` (`echo-area-abort`)
: Cancel or quit the current operation. If completion is being read, this command discards the text of the input line which does not match any completion. If the input line is empty, it aborts the calling function.

`RET` (`echo-area-newline`)
: Accept (or forces completion of) the current input line.

`C-q` (`echo-area-quoted-insert`)
: Insert the next character verbatim. This is how you can insert control characters into a search string.

`M-TAB` (`echo-area-tab-insert`)
: Insert a `TAB` character.

`C-t` (`echo-area-transpose-chars`)
: Transpose the characters at the cursor.

printing character
: Insert the character.

The next group of commands deal with *killing*, and *yanking* text. (Sometimes these operations are called *cut* and *paste*, respectively.) For an in-depth discussion, see Section "Killing and Deleting" in *the GNU Emacs Manual*.

`M-d` (`echo-area-kill-word`)
: Kill the word following the cursor.

`M-DEL` (`echo-area-backward-kill-word`)
`M-BS`
: Kill the word preceding the cursor.

  On some keyboards, the '`Backspace`' key is used instead of `DEL`, so `M-Backspace` has the same effect as `M-DEL`.

`C-k` (`echo-area-kill-line`)
: Kill the text from the cursor to the end of the line.

`C-x DEL` (`echo-area-backward-kill-line`)
: Kill the text from the cursor to the beginning of the line.

`C-y` (`echo-area-yank`)
: Yank back the contents of the last kill.

`M-y` (`echo-area-yank-pop`)
: Yank back a previous kill, removing the last yanked text first.

Sometimes when reading input in the echo area, the command that needed input will only accept one of a list of several choices. The choices represent the *possible completions*, and you must respond with one of them. Since there are a limited number of responses you can make, Info allows you to abbreviate what you type, only typing as much of the response as is necessary to uniquely identify it. In addition, you can request Info to fill in as much of the response as is possible; this is called *completion*.

The following commands are available when completing in the echo area:

`TAB` (`echo-area-complete`)
: Insert as much of a completion as is possible. Otherwise, display a window containing a list of the possible completions of what you have typed so far. For example, if the available choices are:

```
bar
foliate
food
forget
```

and you have typed an '`f`', followed by `TAB`, this would result in '`fo`' appearing in the echo area, since all of the choices which begin with '`f`' continue with '`o`'.

Now if you type `TAB` again, Info will pop up a window showing a node called '`*Completions*`' which lists the possible completions like this:

```
3 completions:
foliate         food
forget
```

i.e., all of the choices which begin with '`fo`'.

Now, typing '`l`' followed by '`TAB`' results in '`foliate`' appearing in the echo area, since that is the only choice which begins with '`fol`'.

`ESC C-v` (`echo-area-scroll-completions-window`)
: Scroll the completions window, if that is visible, or the "other" window if not.

10 Printing Nodes

In general, we recommend that you use TEX to format the document and print sections of it, by running `tex` on the Texinfo source file. However, you may wish to print out the contents of a node as a quick reference document for later use, or if you don't have TEX installed. Info provides you with a command for doing this.

M-x `print-node`

Pipe the contents of the current node through the command in the environment variable `INFO_PRINT_COMMAND`. If the variable does not exist, the node is simply piped to `lpr` (on DOS/Windows, the default is to print the node to the local printer device, `PRN`).

The value of `INFO_PRINT_COMMAND` may begin with the '`>`' character, as in '`>/dev/printer`', in which case Info treats the rest as the name of a file or a device. Instead of piping to a command, Info opens the file, writes the node contents, and closes the file, under the assumption that text written to that file will be printed by the underlying OS.

11 Miscellaneous Commands

GNU Info contains several commands which self-document GNU Info:

M-x `describe-command`
: Read the name of an Info command in the echo area and then display a brief description of what that command does.

M-x `describe-key`
: Read a key sequence in the echo area, and then display the name and documentation of the Info command that the key sequence invokes.

M-x `describe-variable`
: Read the name of a variable in the echo area and then display a brief description of what the variable affects.

M-x `where-is`
: Read the name of an Info command in the echo area, and then display a key sequence which can be typed in order to invoke that command.

H (`get-help-window`)
: Create (or Move into) the window displaying `*Help*`, and place a node containing a quick reference card into it. This window displays the most concise information about GNU Info available.

h (`get-info-help-node`)
: Try hard to visit the node `(info)Help`. The Info file `info.texi` distributed with GNU Emacs contains this node. Of course, the file must first be processed with `texi2any`, and then placed into the location of your Info directory.

`=` (`display-file-info`)
: Show information about what's currently being viewed in the echo area: the Info file name, and current line number and percentage within the current node.

M-x `info-version`
: Display the name and version of the currently running Info program.

Here are the commands for creating a numeric argument:

C-u (`universal-argument`)
: Start (or multiply by 4) the current numeric argument. '`C-u`' is a good way to give a small numeric argument to cursor movement or scrolling commands; '`C-u C-v`' scrolls the screen 4 lines, while '`C-u C-u C-n`' moves the cursor down 16 lines. '`C-u`' followed by digit keys sets the numeric argument to the number thus typed: *C-u 1 2 0* sets the argument to 120.

M-1 (`add-digit-to-numeric-arg`)
1, vi-like operation
M-2 ... M-9
2 ... 9, vi-like operation
M-0
0, vi-like operation
: Add the digit value of the invoking key to the current numeric argument. Once Info is reading a numeric argument, you may just type the digits of the argu-

ment, without the Meta prefix. For example, you might give '`C-l`' a numeric argument of 32 by typing:

```
C-u 3 2 C-l
```

or

```
M-3 2 C-l
```

M-- (`add-digit-to-numeric-arg`)

- To make a negative argument, type `-`. Typing `-` alone makes a negative argument with a value of −1. If you continue to type digit or Meta-digit keys after `-`, the result is a negative number produced by those digits.

`-` doesn't work when you type in the echo area, because you need to be able to insert the '`-`' character itself; use *M--* instead, if you need to specify negative arguments in the echo area.

`C-g` is used to abort the reading of a multi-character key sequence, to cancel lengthy operations (such as multi-file searches) and to cancel reading input in the echo area.

C-g (`abort-key`)
Cancel current operation.

The '`q`' command of Info simply quits running Info.

q (`quit`)
C-x C-c Exit GNU Info.

Here are commands affecting the display of nodes:

C-l (`redraw-display`)
Redraw the display from scratch, or shift the line containing the cursor to a specified location. With no numeric argument, '`C-l`' clears the screen, and then redraws its entire contents. Given a numeric argument of *n*, the line containing the cursor is shifted so that it is on the *n*th line of the window.

M-x `set-screen-height`
Read a height value in the echo area and set the height of the displayed screen to that value. For example, if the operating system tells GNU Info that the screen is 60 lines tall, and it is actually only 40 lines tall, you could use this command to tell Info that the operating system is incorrect.

C-x w (`toggle-wrap`)
Toggles the state of line wrapping in the current window. Normally, lines which are longer than the screen width *wrap*, i.e., they are continued on the next line. Lines which wrap have a '\' appearing in the rightmost column of the screen. You can cause such lines to be terminated at the rightmost column by changing the state of line wrapping in the window with *C-x w*. When a line which needs more space than one screen width to display is displayed, a '`$`' appears in the rightmost column of the screen, and the remainder of the line is invisible. When long lines are truncated, the mode line displays the '`$`' character near its left edge.

On MS-DOS/MS-Windows, this command actually tries to change the dimensions of the visible screen to the value you type in the echo area.

Finally, Info provides a way to display footnotes which might be associated with the current node that you are viewing:

`ESC C-f` (`show-footnotes`)
: Show the footnotes (if any) associated with the current node in another window. You can have Info automatically display the footnotes associated with a node when the node is selected by setting the variable `automatic-footnotes`. See Chapter 13 [`automatic-footnotes`], page 27.

12 Invoking Info

GNU Info accepts several options to control the initial node or nodes being viewed, and to specify which directories to search for Info files. Here is a template showing an invocation of GNU Info from the shell:

```
info [option...] [manual] [menu-or-index-item...]
```

Info will look for an entry called *manual* in the directory files, which are named `dir`, that it finds in its search path. The search is case-insensitive and considers substrings. (If *manual* is not given, by default Info displays a composite directory listing, constructed by combining the `dir` files.) A basic example:

```
info coreutils
```

This looks for an entry labelled `coreutils`, or `Coreutils`, etc., and if found, displays the referenced file (e.g., `coreutils.info`) at the location given. `info coreu` will find it too, if there is no better match.

Another example:

```
info ls
```

Assuming the normal `dir` entry for `ls`, this will show the `ls` documentation, which happens to be within the `coreutils` manual rather than a separate manual. The `dir` entries can point to any node within a manual, so that users don't have to be concerned with the exact structure used by different authors.

If no entry is found in the directories, Info looks for a file called *manual* in its search path. If not found, Info looks for it with the file extensions `.info`, `-info`, and `.inf`. For each of these known extensions, if a regular file is not found, Info looks for a compressed file.[1]

You can specify the name of a node to visit with the `--node` or `-n` option. Alternatively, you can specify the file and node together using the same format that occurs in Info cross-references. These two examples both load the '`Files`' node within the '`emacs`' manual:

```
info emacs -n Files
info '(emacs)Files'
```

If you want to load a file without looking in the search path, specify *manual* either as an absolute path, or as a path relative to the current directory which contains at least one slash character. (You can also use the `--file` option for similar behavior, described below.) Examples:

```
info /usr/local/share/info/bash.info
info ./document.info
```

Info looks for *manual* only in the explicitly specified directory, and adds that directory to its search path.

---

[1] Info supports files compressed with `gzip`, `xz`, `bzip2`, `lzip`, `lzma`, `compress` and `yabba` programs, assumed to have extensions `.z`, `.gz`, `.xz`, `.bz2`, `.lz`, `.lzma`, `.Z`, and `.Y` respectively.

On MS-DOS, Info allows for the Info extension, such as `.inf`, and the short compressed file extensions, such as `.z` and `.gz`, to be merged into a single extension, since DOS doesn't allow more than a single dot in the basename of a file. Thus, on MS-DOS, if Info looks for `bison`, file names like `bison.igz` and `bison.inz` will be found and decompressed by `gunzip`.

Info treats any remaining arguments as the names of menu items, or (see below) index entries. The first argument is a menu item in the '`Top`' node of the file loaded, the second argument is a menu item in the first argument's node, etc. You can move to the node of your choice by specifying the menu names which describe the path to that node. For example,

```
info emacs buffers
info texinfo Overview 'Using Texinfo'
```

The first example selects the menu item '`Buffers`' in the node '`(emacs)Top`'. The second example loads the `texinfo` file and looks in its top-level menu for a '`Overview`' item, looks in the menu of the node referenced, and finally displays the node referenced by the '`Using Texinfo`' item.

If there was only one *menu-or-index-item* argument and it wasn't found as a menu item, Info looks for it as an index entry. For example:

```
info libc printf
```

This loads the libc Info manual and first looks for `printf` in the top-level menu as usual; since it isn't there (at this writing), it then looks in the indices. If it's found there (which it is), the relevant node at the given location is displayed.

If Info is invoked when its standard output is not a terminal, it does not attempt to start an interactive session; rather, it writes the contents of the loaded nodes and subnodes to standard output, as if the `--output=-` and `--subnodes` options were given. This can be used to pipe the contents of Info nodes to another program, such as a pager.

A complete list of options follows.

`--all`

`-a` Find all files matching *manual*. Three usage patterns are supported, as follows.

First, if `--all` is used together with `--where`, `info` prints the names of all matching files found on standard output (including '`*manpages*`' if relevant) and exits.

Second, if `--all` is used together with `--output`, the contents of all matched files are dumped to the specified output file.

Otherwise, an interactive session is initiated. If more than one file matches, a menu node is displayed listing the matches and allowing you to select one. This menu node can be brought back at any time by pressing `C-x f`. If there is only one match, `info` starts as usual.

When used with the `--index-search` option, `info` displays a menu of matching index entries (just as the `virtual-index` command does; see Chapter 7 [Index Commands], page 10).

The `--node` option cannot be used together with this option.

`--apropos=`*`string`*

`-k` *`string`* Specify a string to search in every index of every Info file installed on your system. Info looks up the named *string* in all the indices it can find, prints the results to standard output, and then exits. If you are not sure which Info file explains certain issues, this option is your friend. (If your system has a lot of Info files installed, searching all of them might take some time!)

You can invoke the apropos command from inside Info; see Chapter 6 [Searching Commands], page 8.

`--debug=`*`number`*

`-x` *`number`* Print additional debugging information. The argument specifies the verbosity level, so a higher level includes all the information from lower levels. For all available debugging output, use `-x -1`. Info version 7.1.1 has these levels:

`1` Print information about file handling, such as looking for `dir` files and nodes written with '`--output`'.

`2` Print operations relating to `INFOPATH`.

`3` Print information about node searching.

Debugging output goes to standard error.

`--directory` *`directory-path`*

`-d` *`directory-path`*

Add *directory-path* to the list of directory paths searched when Info needs to find a file. You may issue `--directory` multiple times; once for each directory which contains Info files, or with a list of such directories separated by a colon (or semicolon on MS-DOS/MS-Windows).

Directories specified in the environment variable `INFOPATH` are added to the directories specified with `--directory`, if any. The value of `INFOPATH` is a list of directories usually separated by a colon; on MS-DOS/MS-Windows systems, the semicolon is used. If the value of `INFOPATH` ends with a colon (or semicolon on MS-DOS/MS-Windows), the initial list of directories is constructed by appending the build-time default to the value of `INFOPATH`.

If you do not define `INFOPATH`, Info uses a default path defined when Info was built as the initial list of directories.

Regardless of whether `INFOPATH` is defined, the default documentation directory defined when Info was built is added to the search path. If you do not want this directory to be included, set the `infopath-no-defaults` variable to `On` (see [infopath-no-defaults], page 28).

If the list of directories contains the element `PATH`, that element is replaced by a list of directories derived from the value of the environment variable `PATH`. Each path element of the form *dir*`/base` is replaced by *dir*`/share/info` or *dir*`/info`, provided that directory exists.

`--dribble=`*`file`*

Specify a file where all user keystrokes will be recorded. This file can be used later to replay the same sequence of commands, see the '`--restore`' option below.

`--file` *`manual`*

`-f` *`manual`* Specify a particular manual to visit without looking its name up in any `dir` files.

With this option, it starts by trying to visit `(`*`manual`*`)Top`, i.e., the `Top` node in (typically) *`manual`*`.info`. As above, it tries various file extensions to find the

file. If no such file (or node) can be found, Info exits without doing anything. As with the `dir` lookup described above, any extra *menu-or-index-item* arguments are used to locate a node within the loaded file.

If *manual* is an absolute file name, or begins with `./` or `../`, or contains an intermediate directory, Info will only look for the file in the directory specified, and add this directory to `INFOPATH`. (This is the same as what happens when `--file` is not given.)

`--help`
`-h` Output a brief description of the available Info command-line options.

`--index-search` *`string`*
After processing all command-line arguments, go to the index in the selected Info file and search for index entries which match *string*. If such an entry is found, the Info session begins with displaying the node pointed to by the first matching index entry; press `,` to step through the rest of the matching entries. If no such entry exists, print '`no entries found`' and exit with nonzero status. This can be used from another program as a way to provide online help, or as a quick way of starting to read an Info file at a certain node when you don't know the exact name of that node.

When used with the `--all` option, `info` displays a menu of matching index entries (just as the `virtual-index` command does; see Chapter 7 [Index Commands], page 10).

This command can also be invoked from inside Info; see Chapter 6 [Searching Commands], page 8.

`--init-file` *`INIT-FILE`*
Read key bindings and variable settings from *INIT-FILE* instead of the `.infokey` file in your home directory. See Chapter 15 [Custom Key Bindings], page 33.

`--node` *`nodename`*
`-n` *`nodename`*
Specify a particular node to visit in the initial file that Info loads. You may specify `--node` multiple times: for an interactive Info, each *nodename* is visited in its own window; for a non-interactive Info (such as when `--output` is given) each *nodename* is processed sequentially.

You can specify both the file and node to the `--node` option using the usual Info syntax, but don't forget to escape the open and close parentheses and whitespace from the shell; for example:

```
info --node "(emacs)Buffers"
```

`--output` *`file`*
`-o` *`file`* Direct output to *file*. Each node that Info visits will be output to *file* instead of interactively viewed. A value of `-` for *file* means standard output.

`--no-raw-escapes`
`--raw-escapes, -R`
By default, Info passes SGR terminal control sequences (also known as ANSI escape sequences) found in documents directly through to the terminal. If

you use the `--no-raw-escapes` options, these sequences are displayed as other control characters are; for example, an `ESC` byte is displayed as '`^[`'. The `--raw-escapes` and `-R` options do not do anything, but are included for completeness.

`--restore=`*dribble-file*
: Read keystrokes from *dribble-file*, presumably recorded during previous Info session (see the description of the '`--dribble`' option above). When the keystrokes in the files are all read, Info reverts its input to the usual interactive operation.

`--show-malformed-multibytes`
`--no-show-malformed-multibytes`
: Show malformed multibyte sequences in the output. By default, such sequences are dropped.

`--show-options`
`--usage`
`-O`
: Look for the node that describes how to invoke the program. The name of the program is taken from the other non-option arguments on the command line. For example, '`info emacs -O`' loads the `Emacs Invocation` node of the `emacs` manual.

  This option is provided to make it easier to find the most important usage information in a manual without navigating through menu hierarchies. The effect is similar to the *`M-x`* `goto-invocation` command (see [goto-invocation], page 3) from inside Info.

`--speech-friendly`
`-b`
: On MS-DOS/MS-Windows only, this option causes Info to use standard file I/O functions for screen writes. (By default, Info uses direct writes to the video memory on these systems, for faster operation and colored display support.) This allows the speech synthesizers used by blind persons to catch the output and convert it to audible speech.

`--strict-node-location`
: This option causes Info not to search "nearby" to locate nodes, and instead strictly use the information provided in the Info file. The practical use for this option is for debugging programs that write Info files, to check that they are outputting the correct locations. Due to bugs and malfeasances in the various Info writing programs over the years and versions, it is not advisable to ever use this option when just trying to read documentation.

`--subnodes`
: This option only has meaning when given in conjunction with `--output`. It means to recursively output the nodes appearing in the menus of each node being output. Menu items which resolve to external Info files are not output, and neither are menu items which are members of an index. Each node is only output once.

`-v` *name*`=`*value*
`--variable=`*name*`=`*value*
: Set the `info` variable *name* to *value*. See Chapter 13 [Variables], page 27.

`--version`
: Prints the version information of Info and exits.

`--vi-keys`
: This option binds functions to keys differently, to emulate the key bindings of `vi` and Less. The bindings activated by this option are documented in Section 15.1 [`infokey` format], page 33. (See Chapter 15 [Custom Key Bindings], page 33, for a more general way of altering GNU Info's key bindings.)

`--where`
`--location`
`-w`
: Show the filename that would be read and exit, instead of actually reading it and starting Info.

Finally, Info defines many default key bindings and variables. See Chapter 15 [Custom Key Bindings], page 33, for information on how to customize these settings.

13 Manipulating Variables

GNU Info uses several internal *variables* whose values are looked at by various Info commands. You can change the values of these variables, and thus change the behavior of Info, if desired.

There are three ways to set the value of a variable, listed here in order of precedence:

1. interactively, using the `set-variable` command described below;
2. on the command line, using the `-v` (`--variable`) command line option (see [variable-assignment], page 25);
3. in the `#var` section of the `.infokey` file (see Chapter 15 [Custom Key Bindings], page 33).

M-x `set-variable`
: Read the name of a variable, and the value for it, in the echo area and then set the variable to that value. Completion is available when reading the variable name (see Section 9.3 [The Echo Area], page 14); completion is also available when reading the value when that makes sense.

M-x `describe-variable`
: Read the name of a variable in the echo area and display its value and a brief description.

Here is a list of the variables that you can set in Info.

`automatic-footnotes`
: When set to `On`, footnotes appear and disappear automatically; else, they appear at the bottom of the node text. This variable is `Off` by default. When a node is selected, a window containing the footnotes which appear in that node is created, and the footnotes are displayed within the new window. The window that Info creates to contain the footnotes is called `*Footnotes*`. If a node is selected which contains no footnotes, and a `*Footnotes*` window is on the screen, the `*Footnotes*` window is deleted. Footnote windows created in this fashion are not automatically tiled so that they can use as little of the display as is possible.

`automatic-tiling`
: When set to `On`, creating or deleting a window resizes other windows. This variable is `Off` by default. Normally, typing '`C-x 2`' divides the current window into two equal parts. When `automatic-tiling` is set to `On`, all of the windows are resized automatically, keeping an equal number of lines visible in each window. Any `*Completions*` and `*Footnotes*` windows are exceptions to the automatic tiling; they retain their original size.

`cursor-movement-scrolls`
: When set to `On`, when cursor movement commands reach the top or bottom of a node, another node is loaded depending on the value of `scroll-behavior` (see below). This is the default. When this variable is set to `Off`, cursor movements stop at the top or bottom of a node.

`errors-ring-bell`
: When set to `On` (the default), errors cause the bell to ring.

`follow-strategy`
: When set to `remain` (the default), Info tries to remain within the directory containing the currently displayed Info file when following a cross-reference to an external manual, before looking for the referenced manual in the search path. The alternative value is `path`, which means to look through the search path right away.

  `remain` is intended to be useful for several Texinfo manuals that all reference each other and whose versions should match each other. (For example, various manuals relating to a particular version of Emacs.)

  The alternative behavior, with `path`, may be useful when your Info file search path parallels your command shell's search path, and you always want to find documentation of the version of the program that the shell would execute.

`gc-compressed-files`
: When set to `On`, Info garbage collects files which had to be uncompressed. The default value of this variable is `Off`. Whenever a node is visited in Info, the Info file containing that node is read into memory, and Info reads information about the tags and nodes contained in that file. Once the tags information is read by Info, it is never forgotten. However, the actual text of the nodes does not need to be retained unless a particular Info window needs it. For non-compressed files, node text is not remembered when it is no longer in use. But de-compressing a file can be a time-consuming operation, and so Info tries hard not to do it twice. This variable tells Info it is okay to garbage collect the text of the nodes of a file which was compressed on disk.

`hide-note-references`
: By default, Info displays the contents of Info files mostly verbatim, including text that is used by Info readers for navigation (for example, marking the location of menus or cross-references). If you set this variable to `On`, some of this text is hidden, in a similar way to the `Info-hide-note-references` variable in Emacs (see Section "Emacs Info Variables" in *Info*).

`highlight-searches`
: When set to `On`, highlight matches from searching commands (see Chapter 6 [Searching Commands], page 8).

`infopath-no-defaults`
: Used in conjunction with the `INFOPATH` environment variable (see [INFOPATH], page 23). When set to `On`, the default documentation directory defined when Info was built (e.g., `/usr/share/info`) is not added to the search path for Info files.

`ISO-Latin`
: The default is `On`, which means that Info accepts and displays characters represented by bytes with values 128 and above, such as characters in the UTF-8 encoding or in various 8-bit ISO Latin characters, as well as allowing you to input such characters.

The only reason to set this variable to `Off` would be if your terminal set the eighth bit of a byte to represent the Meta key being pressed.

`key-time` Length of time in milliseconds to wait for the next byte of a byte sequence generated by a key (or key chord) on the keyboard. For example, if the *down* key generates the byte sequence `ESC O B`, and the two bytes `ESC O` have been received, then a `B` byte would have to be received within this length of time for a key press of *down* to be registered. You may wish to set this variable to a larger value for slow terminals or network connections.

If you set this variable to 0, it's unspecified whether a recognized byte sequence representing a key takes precedence over another recognized sequence representing a key that is an initial subsequence of the first sequence. In some cases, you may be able to make pressing a special key on the keyboard that Info doesn't know about (for example, a function key) cause a command to be executed by setting this variable to 0, and giving the byte sequence the key sends in `.infokey`. (See Chapter 15 [Custom Key Bindings], page 33.)

`min-search-length`
Minimum length of a search string (default 1). Attempts to initiate a search for a string (or regular expression) shorter than this value, result in an error.

`mouse` What method to use to get input from a mouse device. The default value is '`Off`'. Set this variable to `normal-tracking` to make Info use "normal tracking mode" if it detects that the terminal supports it. This enables you to scroll the contents of the active window with a mouse scrollwheel.

On terminal emulators running under the X Window System, such as `xterm`, you can usually select text with the mouse. However, mouse tracking mode may interfere with this. When this happens, you may be able to select text by holding down the *Shift* key while clicking and dragging.

`nodeline` How to print the node header line that appears at the top of each node. By default only the pointers to neighbouring nodes are displayed (the "Next", "Prev", and "Up" pointers): this corresponds to the `pointers` value for this variable. To print the entire line, set `nodeline` to the value `print`, which will include the filename and name of the node. To not display the header line at all, use the value `no`.

`scroll-behavior`
`scroll-behaviour`
The two variable names are synonymous. Control what happens when scrolling commands are used at the end or beginning of a node (see Chapter 5 [Scrolling Commands], page 6). The default value for this variable is `Continuous`. Possible values:

`Continuous`
Try to get the first item in this node's menu, or failing that, the '`Next`' node, or failing that, the '`Next`' of the '`Up`' node. This behavior is identical to using the '`]`' (`global-next-node`) and '`[`' (`global-prev-node`) commands.

`Next Only` Only try to get the '`Next`' node.

`Page Only` Just stop, changing nothing. With this value, no scrolling command can change the node that is being viewed.

This variable also affects cursor movement commands (see Chapter 4 [Cursor Commands], page 5) unless the `cursor-movement-scrolls` variable is set to `Off`. See [cursor-movement-scrolls], page 27.

`scroll-last-node`
Control what happens when a scrolling command is issued at the end of the last node. Possible values are:

`Stop` Do not scroll. Display the '`No more nodes within this document`' message. This is the default.

`Top` Go to the top node of the document.

This variable is in effect only if `scroll-behavior` is set to `Continuous`.

`scroll-step`
The number of lines to scroll to bring the cursor back into the window. The default value of this variable is 1, which causes a kind of "smooth scrolling" which some people prefer. Scrolling happens automatically if the cursor has moved out of the visible portion of the node text.

If the variable `scroll-step` is 0, the cursor (and the text it is attached to) is placed in the centre of the window.

`search-skip-screen`
Set the starting point of repeated searches (see [repeated-search], page 8). When set to `Off` (the default), repeated searches start at the position immediately following (when searching in forward direction), or immediately preceding (when searching backwards) the cursor. When set to `On`, repeated searches omit lines visibly displayed on the screen. In other words, forward searches (*}*) start at the beginning of the next page, and backward searches (*{*) start at the end of the previous page.

`show-index-match`
When set to `On` (the default), the portion of the matched search string that you typed is indicated (by displaying it in the "opposite" case) in the result message (see Chapter 6 [`next-index-match`], page 8).

`visible-bell`
When set to `On`, Info attempts to flash the screen instead of ringing the bell. This variable is `Off` by default. If the terminal does not allow flashing, this variable has no effect. (But you can still make Info perform quietly by setting the `errors-ring-bell` variable to `Off`; or using an external command to mute the bell, e.g., `xset b 0 0 0`.)

14 Colors and Styles

You can choose to highlight parts of Info's display, such as cross-references and search matches, using a variety of styles, including colors, boldface and underline. Here are the variables that are available to do this:

`link-style`
: Used for cross-references and menu entries.

`active-link-style`
: Used for a cross-reference or menu entry when typing `RET` would have the effect of following said cross-reference or menu entry.

`match-style`
: Used for matches from a search command. (See Chapter 6 [Searching Commands], page 8.)

Each of these is given in the `.infokey` file just as the variables in the previous chapter. Their values are a comma-separated list of values in the following table:

`black`
`red`
`green`
`yellow`
`blue`
`magenta`
`cyan`
`white`
: Use the color specified for text.

`nocolor`
`nocolour`
: Turn off any color that was in effect, using the terminal's default color.

`bgblack`
`bgred`
`bggreen`
`bgyellow`
`bgblue`
`bgmagenta`
`bgcyan`
`bgwhite`
: Use the color specified for the background.

`bgnocolor`
`bgnocolour`
: Use the terminal's default background color.

`underline`
`nounderline`
: Turn text underline on or off.

`standout`
`nostandout`
: Turn 'standout mode' on or off. Standout mode entails the use of appearance modes that make text stand out, and varies between terminals.

`bold`
`regular`
`nobold` Turn boldface on or off.

`blink`
`noblink` Make the text blink, or not.

Here is an sample excerpt from an `.infokey` file:

```
#var
link-style=yellow
active-link-style=yellow,bold
match-style=underline,bold,nocolor
```

With this, cross-references are all yellow, and active cross-references are additionally displayed in bold. Any search matches will be shown in bold, and underlined. Moreover, if there is a search match inside a cross-reference, the '`nocolor`' rendition style will cancel the yellow color, leaving the text in the match the terminal's default color. (Note, however, that the rendition styles for active cross-references take priority over those for search matches, so search matches there will still be displayed in yellow.)

15 Customizing Key Bindings and Variables

Info allows you to override the default key-to-command bindings and variable settings described in this document. (The `--vi-keys` option rebinds many keys at once; see [–vi-keys], page 26.)

On startup, GNU Info looks for a configuration file in the invoker's `HOME` directory called `.infokey`, i.e., `~/.infokey`.[1] If it is present, then Info adopts the key bindings and variable settings contained therein. To use an alternative configuration file, use the `--init-file` option (see [–init-file], page 24).

Variables may also be set on the command line with the `--variable` option (see [variable-assignment], page 25). Variable settings on the command line override settings from the `.infokey` file.

15.1 `infokey` format

Here is an example `.infokey` file which specifies the key bindings that are activated by the `--vi-keys` option to Info (see [–vi-keys], page 26).

```
#info
g       first-node
G       last-node
\mb     beginning-of-node
\me     end-of-node
j       next-line
k       prev-line

f       scroll-forward-page-only
^f      scroll-forward-page-only
\m\     scroll-forward-page-only
z       scroll-forward-page-only-set-window

b       scroll-backward-page-only
^b      scroll-backward-page-only
w       scroll-backward-page-only-set-window

\kd     down-line
^e      down-line
^j      down-line
^m      down-line
\ku     up-line
^y      up-line
^k      up-line

d       scroll-half-screen-down
^d      scroll-half-screen-down
```

[1] Due to the limitations of DOS filesystems, the MS-DOS version of Info looks for a file `_infokey` instead. If the `HOME` variable is not defined, Info additionally looks in the current directory.

```
u       scroll-half-screen-up
^u      scroll-half-screen-up

^xn     next-node
^xp     prev-node
^xu     up-node
'       last-node
\mt     top-node
\md     dir-node

^xg     goto-node
I       goto-invocation-node

n       search-next
N       search-previous

\mf     xref-item
^xr     xref-item

\mg     select-reference-this-line
^x^j    select-reference-this-line
^x^m    select-reference-this-line

^c      abort-key

\mh     get-info-help-node

:q      quit
ZZ      quit

#echo-area
\mh     echo-area-backward
\ml     echo-area-forward
\m0     echo-area-beg-of-line
\m$     echo-area-end-of-line
\mw     echo-area-forward-word
\mx     echo-area-delete
\mu     echo-area-abort
^v      echo-area-quoted-insert
\mX     echo-area-kill-word
```

The file consists of one or more *sections*. Each section starts with a line that identifies the type of section. The possible sections are:

`#info` Key bindings for Info windows. The start of this section is indicated by a line containing just `#info` by itself. If this is the first section in the source file, the `#info` line can be omitted. The rest of this section consists of lines of the form:

string `whitespace` *action* `[ whitespace [ # comment ] ] newline`

Whitespace is any sequence of one or more spaces and/or tabs. Comment is any sequence of any characters, excluding newline. *string* is the key sequence which invokes the action. *action* is the name of an Info command. The characters in *string* are interpreted literally or prefixed by a caret (`^`) to indicate a control character. A backslash followed by certain characters specifies input keystrokes as follows:

`\b`	Backspace
`\e`	Escape (ESC)
`\n`	Newline
`\r`	Return
`\t`	Tab
`\ku`	Up arrow
`\kd`	Down arrow
`\kl`	Left arrow
`\kr`	Right arrow
`\kU`	Page Up
`\kD`	Page Down
`\kh`	HOME
`\ke`	END
`\kx`	Delete (DEL)
`\mx`	Meta-*x* where *x* is any character as described above.

Backslash followed by any other character indicates that character is to be taken literally. Characters which must be preceded by a backslash include caret, space, tab, and backslash itself.

`#echo-area`
: Key bindings for the echo area. The start of this section is indicated by a line containing just `#echo-area` by itself. The rest of this section has a syntax identical to that for the key definitions for the Info area, described above.

`#var`
: Variable initializations. The start of this section is indicated by a line containing just `#var` by itself. Following this line is a list of variable assignments, one per line. Each line consists of a variable name (see Chapter 13 [Variables], page 27) followed by `=` followed by a value. There may be no white space between the variable name and the `=`, and all characters following the `=`, including white space, are included in the value.

Blank lines and lines starting with `#` are ignored, except for the special section header lines.

Key bindings defined in the `.infokey` file take precedence over GNU Info's default key bindings, whether or not '`--vi-keys`' is used. A default key binding may be disabled by

overriding it in the `.infokey` file with the action `invalid`. In addition, *all* default key bindings can be disabled by adding this line *anywhere* in the relevant section:

```
#stop
```

This will cause GNU Info to ignore all the default key commands for that section.

Beware: `#stop` can be dangerous. Since it disables all default key bindings, you must supply enough new key bindings to enable all necessary actions. Failure to bind any key to the `quit` command, for example, can lead to frustration.

Note that some keys (such as `C-c`) have special meanings to terminals, and any bindings for these would not be effective. See Section "Special Characters" in *GNU Coreutils*.

The order in which key bindings are defined in the `.infokey` file is not important, except that the command summary produced by the `get-help-window` command only displays the *first* key that is bound to each command.

16 Alternate MS-DOS/Windows keybindings

Here are some keybindings that are only used in Info on MS-DOS/Windows.

Cursor commands

`Home` `beginning-of-line`

`End` `end-of-line`

C`-RIGHT` `forward-word`

C`-LEFT` `backward-word`

C`-Home` `beginning-of-node`

C`-End` `end-of-node`

Node commands

C`-NEXT` `next-node`

C`-PREVIOUS`
`prev-node`

C`-UP` *(an arrow key)*
`up-node`

C`-CENTER` `history-node`

The `NEXT` key is known as the `PgDn` key on some keyboards. The `PREVIOUS` key is known as the `PgUp` key on some keyboards.

Echo area commands

C`-RIGHT` `echo-area-forward-word`

C`-LEFT` `echo-area-backward-word`

Shift`-TAB`
`echo-area-tab-insert`

On DOS/Windows only, the *Shift*`-TAB` key is an alias for *M*`-TAB`. This key is sometimes called '`BackTab`'.

Miscellaneous commands

`F1` `get-help-window`

Appendix A Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X