GNU Online Documentation

Reading and Making Info files

For GNU Info version 2.9

Brian J. Fox (bfox@ai.mit.edu)

Edited by Cygnus Support (doc@cygnus.com)

Table of Contents

Info

GNU Online Documentation

You can read the manuals for GNU software either on paper, as with any other manual, or as online *info files*, using an ordinary ASCII terminal.

You can browse through the online documentation with GNU Emacs, or with the program Info, a smaller program intended just for the purpose of viewing info files. This manual describes version 2.9 of Info.

Info files are generated by the program `makeinfo` from a *texinfo* source file. (Texinfo is a documentation markup language designed to allow the same source file to generate either printed or online documentation.) This manual describes version 1.51 of Makeinfo. The Texinfo language is described in a separate manual, *Texinfo: The GNU Documentation Format*.

1 Reading GNU Online Documentation

The info file version of a manual is organized into *nodes*, which usually correspond to the chapters and sections of the printed book. You can follow them in sequence, if you wish, just like in the printed book—but there are also other choices. Info files have menus that let you go quickly to the node that has the information you need. Info has “hot” references; if one section refers to another, you can tell Info to take you immediately to that other section—and you can get back again easily to take up your reading where you left off. Naturally, you can also search for particular words or phrases.

The best way to get started with the online documentation system is to use a programmed tutorial by running Info itself. You can get into Info by just typing its name—no options or arguments are necessary—at your shell’s prompt (shown as ‘eg$’ here):

```
eg$ info
```

Info displays its first screen, a menu of the documentation available, and awaits your input. Type the single letter

```
h
```

to request a tutorial, designed to teach you how to use Info.

If you already use Emacs, you may want to get into the documentation browsing mode, instead, by typing `C-h i` inside Emacs.

You can get out of Info at any time by typing the single letter `q`.

Info can also display a summary of all its commands at any time, when you type the single character `?`.

What is Info?

This text documents the use of the GNU Info program, version 2.9.

Info is a program which is used to view info files on an ASCII terminal. *info files* are the result of processing texinfo files with the program `makeinfo` or with the Emacs command `M-x texinfo-format-buffer`. Finally, *texinfo* is a documentation language which allows a printed manual and online documentation (an info file) to be produced from a single source file.

1.1 Command Line Options

GNU Info accepts several options to control the initial node 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-name option-value] menu-item...
```

The following *option-names* are available when invoking Info from the shell:

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

Adds *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. Alternatively, you may specify a value for the environment variable `INFOPATH`; if `--directory` is not given, the value of `INFOPATH` is used. The value of `INFOPATH` is a colon separated list of directory names. If you do not supply `INFOPATH` or `--directory-path` a default path is used.

`--file` *filename*
`-f` *filename*

Specifies a particular info file to visit. Instead of visiting the file `dir`, Info will start with `(`*filename*`)Top` as the first file and node.

`--node` *nodename*
`-n` *nodename*

Specifies a particular node to visit in the initial file loaded. This is especially useful in conjunction with `--file`[1]. 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.

`--output` *filename*
`-o` *filename*

Specify *filename* as the name of a file to output to. Each node that Info visits will be output to *filename* instead of interactively viewed. A value of `-` for *filename* specifies the standard output.

`--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.

[1] Of course, you can specify both the file and node in a `--node` command; but don't forget to escape the open and close parentheses from the shell as in: `info --node '(emacs)Buffers'`

`--help`
`-h`
: Produces a relatively brief description of the available Info options.

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

`menu-item`
: Remaining arguments to Info are treated as the names of menu items. The first argument would be a menu item in the initial node visited, while the second argument would be a menu item in the first argument's node. You can easily move to the node of your choice by specifying the menu names which describe the path to that node. For example,

```
info emacs buffers
```

first selects the menu item '`Emacs`' in the node '`(dir)Top`', and then selects the menu item '`Buffers`' in the node '`(emacs)Top`'.

1.2 Moving the Cursor

Many people find that reading screens of text page by page is made easier when one is able to indicate particular pieces of text with some kind of pointing device. Since this is the case, GNU Info (both the Emacs and standalone versions) have several commands which allow you to move the cursor about the screen. The notation used in this manual to describe keystrokes is identical to the notation used within the Emacs manual, and the GNU Readline manual. See section "Character Conventions" in *the GNU Emacs Manual*, if you are unfamilar with the notation.

The following table lists the basic cursor movement commands in Info. Each entry consists of the key sequence you should type to execute the cursor movement, the `M-x`[2] command name (displayed in parentheses), and a short description of what the command does. All of the cursor motion commands can take an *numeric* argument (see Section 1.9 "Miscellaneous Commands," page 17), to find out how to supply them. 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.

[2] `M-x` is also a command; it invokes `execute-extended-command`. See section "Executing an extended command" in *the GNU Emacs Manual*, for more detailed information.

`C-n` (`next-line`)
: Moves the cursor down to the next line.

`C-p` (`prev-line`)
: 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`)
: Moves the cursor to the end of the current line.

`C-f` (`forward-char`)
: Move the cursor forward a character.

`C-b` (`backward-char`)
: Move the cursor backward a character.

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

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

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

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

`M-r` (`move-to-window-line`)
: Moves 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.

1.3 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`)
`C-v`
: 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.

DEL (scroll-backward)
M-v Shift the text in this window down. The inverse of scroll-forward.

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-behaviour. See Section 1.10 “Manipulating Variables,” page 19, for more information.

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.

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.

1.4 Selecting a New Node

This section details the numerous Info commands which select a new node to view in the current window.

The most basic node commands are ‘n’, ‘p’, ‘u’, and ‘l’.

When you are viewing a node, the top line of the node contains some Info *pointers* which describe where the next, previous, and up nodes are. Info uses this line to move about the node structure of the file when you use the following commands:

n (next-node)
: Selects the 'Next' node.

p (prev-node)
: Selects the 'Prev' node.

u (up-node)
: Selects the 'Up' node.

You can easily select a node that you have already viewed in this window by using the 'l' command – this name stands for "last", and actually moves through the list of already visited nodes for this window. 'l' with a negative numeric argument moves forward through the history of nodes for this window, so you can quickly step between two adjacent (in viewing history) nodes.

l (history-node)
: Selects the most recently selected node in this window.

Two additional commands make it easy to select the most commonly selected nodes; they are 't' and 'd'.

t (top-node)
: Selects the node 'Top' in the current info file.

d (dir-node)
: Selects the directory node (i.e., the node '(dir)').

Here are some other commands which immediately result in the selection of a different node in the current window:

< (first-node)
: Selects the first node which appears in this file. This node is most often 'Top', but it doesn't have to be.

> (last-node)
: Selects the last node which appears in this file.

] (global-next-node)
: Moves forward or down through node structure. If the node that you are currently viewing has a 'Next' pointer, that node is selected. Otherwise, if this node has a menu, the first menu item is selected. If there is no 'Next' and no menu, the same process is tried with the 'Up' node of this node.

[(global-prev-node)
: Moves backward or up through 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 behaviour as `global-next-node` and `global-prev-node` while simply scrolling through the file with SPC and DEL; See Section 1.10 “`scroll-behaviour`,” page 19, for more information.

`g` (`goto-node`)
: Reads the name of a node and selects it. No completion is done while reading the node name, since the desired node may reside in a separate file. The node must be typed exactly as it appears in the info file. A file name may be included as with any node specification, for example

```
g(emacs)Buffers
```

finds the node ‘`Buffers`’ in the info file ‘`emacs`’.

`C-x k` (`kill-node`)
: Kills a node. The node name is prompted for in the echo area, with a default of the current node. *Killing* a node means that Info tries hard to forget about it, removing it from the list of history nodes kept for the window where that node is found. Another node is selected in the window which contained the killed node.

`C-x C-f` (`view-file`)
: Reads 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`)
: Makes 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`)
: Selects 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.

1.5 Searching an Info File

GNU Info allows you to search for a sequence of characters throughout an entire info file, search through the indices of an info file, or find areas within an info file which discuss a particular topic.

`s` (`search`)
: Reads a string in the echo area and searches for it.

`C-s` (`isearch-forward`)
: Interactively searches forward through the info file for a string as you type it.

`C-r` (`isearch-backward`)
: Interactively searches backward through the info file for a string as you type it.

`i` (`index-search`)
: Looks up a string in the indices for this info file, and selects a node where the found index entry points to.

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

The most basic searching command is ‘`s`’ (`search`). The ‘`s`’ command prompts you for a string in the echo area, and then searches the remainder of the info file for an ocurrence 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 within ‘`[`’ and ‘`]`’; 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.

1.6 Selecting Cross References

We have already discussed the ‘`Next`’, ‘`Prev`’, and ‘`Up`’ pointers which appear at the top of a node. In addition to these pointers, a node may contain other pointers which refer you to a different node, perhaps in another info file. Such pointers are called *cross references*, or *xrefs* for short.

1.6.1 Parts of an Xref

Cross references have two major 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 colon ‘`:`’; first the label appears, and then the target. For example, in the sample menu cross reference below, the single colon separates the label from the target.

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

Note the '.' which 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 references 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 "Writing an Xref" in *the Texinfo Manual*, for more information on creating your own texinfo cross references.

Info

1.6.2 Selecting Xrefs

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

1 (menu-digit)
2 ... 9
: 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.

0 (last-menu-item)
: 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.

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

This table lists the Info commands which operate on note 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.

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

`TAB` (`move-to-next-xref`)
Moves 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`)
Moves the cursor the start of the nearest previous menu item or note reference in this node.

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

1.7 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 modeline. 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.

1.7.1 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. It can also tell you if the indirect tags table for this info file needs to be updated, and whether or not the info file was compressed when stored on disk.

Here is a sample mode line for a window containing an uncompressed file named ‘`dir`’, showing the node ‘`Top`’.

```
-----Info: (dir)Top, 40 lines --Top-------------------------
            ^^   ^   ^^^        ^^
          (file)Node #lines    where
```

When a node comes from a file which is compressed on disk, this is indicated in the mode line with two small 'z''s. In addition, if the info file containing the node has been split into subfiles, the name of the subfile containing the node appears in the modeline as well:

```
--zz-Info: (emacs)Top, 291 lines --Top-- Subfile: emacs-1.Z-
```

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---------------------
```

1.7.2 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 1.7.1 "The Mode Line," page 12) and history of nodes viewed in that window (see Section 1.4 "history-node," page 7).

C-x o (next-window)

Selects 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

Selects the previous window on the screen. This is identical to 'C-x o' with a negative argument.

C-x 2 (split-window)

Splits 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, please see Section 1.10 "automatic-tiling," page 19 for more information.

`C-x 0` (`delete-window`)
: Deletes 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`)
: Deletes all of the windows excepting the current one.

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

`C-x ^` (`grow-window`)
: Grows (or shrinks) the current window. Given a numeric argument, grows the current window that many lines; with a negative numeric argument, the window is shrunk instead.

`C-x t` (`tile-windows`)
: Divides 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 Section 1.10 "`automatic-tiling`," page 19.

1.7.3 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`)
: Moves forward a character.

`C-b` (`echo-area-backward`)
: Moves backward a character.

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

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

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

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

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

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

`C-g` (`echo-area-abort`)
: Cancels or quits the current operation. If completion is being read, ‘`C-g`’ discards the text of the input line which does not match any completion. If the input line is empty, ‘`C-g`’ aborts the calling function.

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

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

printing character (`echo-area-insert`)
: Inserts the character.

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

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

The next group of commands deal with *killing*, and *yanking* text. For an in depth discussion of killing and yanking, see section “Killing and Deleting” in *the GNU Emacs Manual*

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

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

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

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

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

`M-y` (`echo-area-yank-pop`)
: Yanks 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`)
`SPC`
: Inserts as much of a completion as is possible.

`?` (`echo-area-possible-completions`)
: Displays 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 ‘`?`’, the possible completions would contain:

```
foliate
food
forget
```

i.e., all of the choices which begin with ‘`f`’. Pressing `SPC` or `TAB` would result in ‘`fo`’ appearing in the echo area, since all of the choices which begin with ‘`f`’ continue with ‘`o`’. 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`)
: Scrolls the completions window, if that is visible, or the "other" window if not.

1.8 Printing Out Nodes

You may wish to print out the contents of a node as a quick reference document for later use. Info provides you with a command for doing this. 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.

`M-x print-node`
: Pipes the contents of the current node through the command in the environment variable `INFO_PRINT_COMMAND`. If the variable doesn't exist, the node is simply piped to `lpr`.

1.9 Miscellaneous Commands

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

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

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

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

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

`C-h` (`get-help-window`)
`?`
: Creates (or moves into) the window displaying `*Help*`, and places 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`)
: Tries hard to visit the node `(info)Help`. The info file ‘`info.texi`’ distributed with GNU Info contains this node. Of course, the file must first be processed with `makeinfo`, and then placed into the location of your info directory.

Here are the commands for creating a numeric argument:

C-u (universal-argument)
: Starts (or multiplies 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.

M-1 (add-digit-to-numeric-arg)
M-2 ... M-9
: Adds 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 argument, 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
```

'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)
: Cancels current operation.

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

q (quit)
: Exits GNU Info.

If the operating system tells GNU Info that the screen is 60 lines tall, and it is actually only 40 lines tall, here is a way to tell Info that the operating system is correct.

M-x set-screen-height
: Reads a height value in the echo area and sets the height of the displayed screen to that value.

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

ESC C-f (show-footnotes)
: Shows 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 Section 1.10 "automatic-footnotes," page 19.

1.10 Manipulating Variables

GNU Info contains several *variables* whose values are looked at by various Info commands. You can change the values of these variables, and thus change the behaviour of Info to more closely match your environment and info file reading manner.

`M-x set-variable`
: Reads the name of a variable, and the value for it, in the echo area and then sets the variable to that value. Completion is available when reading the variable name; often, completion is available when reading the value to give to the variable, but that depends on the variable itself. If a variable does *not* supply multiple choices to complete over, it expects a numeric value.

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

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

`automatic-footnotes`
: When set to `On`, footnotes appear and disappear automatically. This variable is `On` 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. There are exceptions to the automatic tiling; specifically, the windows '`*Completions*`' and '`*Footnotes*`' are *not* resized through automatic tiling; they remain their original size.

`visible-bell`
: When set to `On`, GNU Info attempts to flash the screen instead of ringing the bell. This variable is `Off` by default. Of course,

Info can only flash the screen if the terminal allows it; in the case that the terminal does not allow it, the setting of this variable has no effect. However, you can make Info perform quietly by setting the `errors-ring-bell` variable to `Off`.

`errors-ring-bell`
: When set to `On`, errors cause the bell to ring. The default setting of this variable is `On`.

`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 core, 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 remain in core unless a particular info window needs it. For non-compressed files, the text of the nodes does not remain in core 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. `gc-compressed-files` tells Info it is okay to garbage collect the text of the nodes of a file which was compressed on disk.

`show-index-match`
: When set to `On`, the portion of the matched search string is highlighted in the message which explains where the matched search string was found. The default value of this variable is `On`. When Info displays the location where an index match was found, (see Section 1.5 “`next-index-match`,” page 9), the portion of the string that you had typed is highlighted by displaying it in the inverse case from its surrounding characters.

`scroll-behaviour`
: Controls what happens when forward scrolling is requested at the end of a node, or when backward scrolling is requested at the beginning of a node. The default value for this variable is `Continuous`. There are three possible values for this variable:

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

`Next Only`
: Only tries to get the ‘`Next`’ node.

`Page Only`
: Simply gives up, changing nothing. If `scroll-behaviour` is `Page Only`, no scrolling command can change the node that is being viewed.

`scroll-step`
: The number of lines to scroll when the cursor moves out of the window. Scrolling happens automatically if the cursor has moved out of the visible portion of the node text when it is time to display. Usually the scrolling is done so as to put the cursor on the center line of the current window. However, if the variable `scroll-step` has a nonzero value, Info attempts to scroll the node text by that many lines; if that is enough to bring the cursor back into the window, that is what is done. The default value of this variable is 0, thus placing the cursor (and the text it is attached to) in the center of the window. Setting this variable to 1 causes a kind of "smooth scrolling" which some people prefer.

`ISO-Latin`
: When set to `On`, Info accepts and displays ISO Latin characters. By default, Info assumes an ASCII character set. `ISO-Latin` tells Info that it is running in an environment where the European standard character set is in use, and allows you to input such characters to Info, as well as display them.

2 Making Info Files from Texinfo Files

Makeinfo is the program that builds info files from texinfo files. Before reading this chapter, you should be familiar with reading info files.

If you want to run Makeinfo on a texinfo file prepared by someone else, this chapter contains all you need to know.

However, to write your own texinfo files, you should also read the Texinfo manual. See section "Texinfo" in *Texinfo—the GNU Documentation Format*.

2.1 Controlling Paragraph Formats

In general, Makeinfo *fills* the paragraphs that it outputs to the info file. Filling is the process of breaking up and connecting lines such that the output is nearly justified. With Makeinfo, you can control:

- The width of each paragraph (the *fill-column*).
- The amount of indentation that the first line of the paragraph receives (the *paragraph-indentation*).

2.2 Command Line Options for Makeinfo

The following command line options are available for Makeinfo.

`-I` *dir*
: Adds *dir* to the directory search list for finding files which are included with the `@include` command. By default, only the current directory is searched.

`-D` *var*
: Defines the texinfo flag *var*. This is equivalent to '`@set` *var*' in the texinfo file.

`-U` *var*
: Makes the texinfo flag *var* undefined. This is equivalent to '`@clear` *var*' in the texinfo file.

`--error-limit` *num*
: Sets the maximum number of errors that Makeinfo will print before exiting (on the assumption that continuing would be useless). The default number of errors printed before Makeinfo gives up on processing the input file is 100.

`--fill-column` *num*
: Specifies the maximum right-hand edge of a line. Paragraphs that are filled will be filled to this width. The default value for `fill-column` is 72.

`--footnote-style` *`style`*
: Sets the footnote style to *`style`*. *`style`* should either be '`separate`' to have Makeinfo create a separate node containing the footnotes which appear in the current node, or '`end`' to have Makeinfo place the footnotes at the end of the current node.

`--no-headers`
: Suppress the generation of menus and node headers. This option is useful together with the '`--output` *`file`*' and '`--no-split`' options (see below) to produce a simple formatted file (suitable for printing on a dumb printer) from texinfo source. If you do not have TEX, these two options may allow you to get readable hard copy.

`--no-split`
: Suppress the splitting stage of Makeinfo. In general, large output files (where the size is greater than 70k bytes) are split into smaller subfiles, each one apporximately 50k bytes. If you specify '`--no-split`', Makeinfo will not split up the output file.

`--no-pointer-validate`
`--no-validate`
: Suppress the validation phase of Makeinfo. Normally, after the file is processed, some consistency checks are made to ensure that cross references can be resolved, etc. See Section 2.3 "What Makes a Valid Info File?," page 25.

`--no-warn`
: Suppress the output of warning messages. This does *not* suppress the output of error messages, simply warnings. You might want this if the file you are creating has texinfo examples in it, and the nodes that are referenced don't actually exist.

`--no-number-footnotes`
: Suppress the automatic numbering of footnotes. The default is to number each footnote sequentially in a single node, resetting the current footnote number to 1 at the start of each node.

`--output` *`file`*
`-o` *`file`*
: Specify that the output should be directed to *`file`* instead of the file name specified in the `@setfilename` command found in the texinfo source. *`file`* can be the special token '`-`', which specifies standard output.

`--paragraph-indent` *num*
: Sets the paragraph indentation to *num*. The value of *num* is interpreted as follows:

  - A value of 0 (or 'none') means not to change the existing indentation (in the source file) at the start of paragraphs.
  - A value less than zero means to indent paragraph starts to column zero by deleting any existing indentation.
  - A value greater than zero is the number of spaces to leave at the front of each paragraph start.

`--reference-limit` *num*
: When a node has many references in a single texinfo file, this may indicate an error in the structure of the file. *num* is the number of times a given node may be referenced (with `@prev`, `@next`, `@note`, or appearing in an `@menu`, for example) before Makeinfo prints a warning message about it.

`--verbose`
: Causes Makeinfo to inform you as to what it is doing. Normally Makeinfo only outputs text if there are errors or warnings.

`--version`
: Displays the Makeinfo version number.

2.3 What Makes a Valid Info File?

If you have not used '`--no-pointer-validate`' to suppress validation, Makeinfo will check the validity of the final info file. Mostly, this means ensuring that nodes you have referenced really exist. Here is a complete list of what is checked:

1. If a node reference such as `Prev`, `Next` or `Up` is a reference to a node in this file (i.e., not an external reference such as '`(DIR)`'), then the referenced node must exist.
2. In a given node, if the node referenced by the `Prev` is different than the node referenced by the `Up`, then the node referenced by the `Prev` must have a `Next` which references this node.
3. Every node except `Top` must have an `Up` field.
4. The node referenced by `Up` must contain a reference to this node, other than a `Next` reference. Obviously, this includes menu items and followed references.
5. If the `Next` reference is not the same as the `Next` reference of the `Up` reference, then the node referenced by `Next` must have a `Prev`

reference pointing back at this node. This rule still allows the last node in a section to point to the first node of the next chapter.

2.4 Defaulting the Prev, Next, and Up

If you write the @node commands in your texinfo source file without Next, Prev, and Up pointers, Makeinfo will fill in the pointers from context (by reference to the menus in your source file).

Although the definition of an info file allows a great deal of flexibility, there are some conventions that you are urged to follow. By letting Makeinfo default the Next, Prev, and Up pointers you can follow these conventions with a minimum of effort.

A common error ocurrs when adding a new node to a menu; often the nodes which are referenced in the menu do not point to each other in the same order as they appear in the menu.

Makeinfo node defaulting helps with this particular problem by not requiring any explicit information beyond adding the new node (so long as you do include it in a menu).

The node to receive the defaulted pointers must be followed immediately by a sectioning command, such as @chapter or @section, and must appear in a menu that is one sectioning level or more above the sectioning level that this node is to have.

Here is an example of how to use this feature.

```
@setfilename default-nodes.info
@node Top
@chapter Introduction
@menu
* foo::  the foo node
* bar::  the bar node
@end menu

@node foo
@section foo
this is the foo node.

@node bar
@section Bar
This is the Bar node.
@bye
```

produces

```
Info file default-nodes.info, produced by Makeinfo, -*- Text -*-

from input file default-nodes.texinfo.

File: default-nodes.info,  Node: Top
```

```
Introduction
************

* Menu:

* foo::  the foo node
* bar::  the bar node

File: default-nodes.info,  Node: foo,  Next: bar,  Up: Top

foo
===

this is the foo node.

File: default-nodes.info,  Node: bar,  Prev: foo,  Up: Top

Bar
===

This is the Bar node.
```

Info

Index