Info

The

On-line, Menu-driven

GNU Documentation System

Published by the Free Software Foundation
675 Massachusetts Avenue,
Cambridge, MA 02139 USA

1 Getting Started

This first part of the Info manual describes how to get around inside of Info. The second part of the manual describes various advanced Info commands, and how to write an Info as distinct from a Texinfo file. The third part is about how to generate Info files from Texinfo files.

This manual is primarily designed for use on a computer, so that you can try Info commands while reading about them. Reading it on paper is less effective, since you must take it on faith that the commands described really do what the manual says. By all means go through this manual now that you have it; but please try going through the on-line version as well.

There are two ways of looking at the online version of this manual:

1. Type `info` at your shell's command line. This approach uses a small stand-alone program designed just to read Info files.
2. Type `emacs` at the command line; then type `C-h i` (Control `h`, followed by `i`). This approach uses the Info mode of the Emacs program, an editor with many other capabilities.

In either case, then type `mInfo` (just the letters), followed by `RET`—the "Return" or "Enter" key. At this point, you should be ready to follow the instructions in this manual as you read them on the screen.

1.1 Starting Info on a Small Screen

(In Info, you only see this section if your terminal has a small number of lines; most readers pass by it without seeing it.)

Since your terminal has an unusually small number of lines on its screen, it is necessary to give you special advice at the beginning.

If you see the text '`--All----`' at near the bottom right corner of the screen, it means the entire text you are looking at fits on the screen. If you see '`--Top----`' instead, it means that there is more text below that does not fit. To move forward through the text and see another screen full, press the Space bar, `SPC`. To move back up, press the key labeled '`Backspace`' or `DELETE`.

1.2 How to use Info

You are talking to the program Info, for reading documentation.

Right now you are looking at one *Node* of Information. A node contains text describing a specific topic at a specific level of detail. This node's topic is "how to use Info".

The top line of a node is its *header*. This node's header (look at it now) says that it is the node named 'Help' in the file 'info'. It says that the 'Next' node after this one is the node called 'Help-P'. An advanced Info command lets you go to any node whose name you know.

Besides a 'Next', a node can have a 'Previous' or an 'Up'. This node has a 'Previous' but no 'Up', as you can see.

Now it is time to move on to the 'Next' node, named 'Help-P'.

>> Type 'n' to move there. Type just one character; do not type the quotes and do not type a RET afterward.

'>>' in the margin means it is really time to try a command.

1.3 Returning to the Previous node

This node is called 'Help-P'. The 'Previous' node, as you see, is 'Help', which is the one you just came from using the n command. Another n command now would take you to the next node, 'Help-^L'.

>> But do not do that yet. First, try the p command, which takes you to the 'Previous' node. When you get there, you can do an n again to return here.

This all probably seems insultingly simple so far, but *do not* be led into skimming. Things will get more complicated soon. Also, do not try a new command until you are told it is time to. Otherwise, you may make Info skip past an important warning that was coming up.

>> Now do an n to get to the node 'Help-^L' and learn more.

1.4 The Space, Delete, B and ^L commands.

This node's header tells you that you are now at node 'Help-^L', and that p would get you back to 'Help-P'. The node's title is underlined; it says what the node is about (most nodes have titles).

This is a big node and it does not all fit on your display screen. You can tell that there is more that is not visible because you can see the string '--Top-----' rather than '--All----' near the bottom right corner of the screen.

The Space, Delete and B commands exist to allow you to "move around" in a node that does not all fit on the screen at once. Space moves forward, to show what was below the bottom of the screen. Delete moves backward, to show what was above the top of the screen (there is not anything above the top until you have typed some spaces).

>> Now try typing a Space (afterward, type a Delete to return here).

When you type the space, the two lines that were at the bottom of the screen appear at the top, followed by more lines. Delete takes the two lines from the top and moves them to the bottom, *usually*, but if there are not a full screen's worth of lines above them they may not make it all the way to the bottom.

If you type Space when there is no more to see, it rings the bell and otherwise does nothing. The same goes for Delete when the header of the node is visible.

If your screen is ever garbaged, you can tell Info to print it out again by typing C-l (Control-L, that is—hold down "Control" and type an L or l).

>> Type C-l now.

To move back to the beginning of the node you are on, you can type a lot of Deletes. You can also type simply b for beginning. >> Try that now. (We have put in enough verbiage to push this past the first screenful, but screens are so big nowadays that perhaps it isn't enough. You may need to shrink your Emacs or Info window.) Then come back, with Spaces.

If your screen is very tall, all of this node might fit at once. In that case, "b" won't do anything. Sorry; what can we do?

You have just learned a considerable number of commands. If you want to use one but have trouble remembering which, you should type a `?` which prints out a brief list of commands. When you are finished looking at the list, make it go away by typing a `SPC`.

>> Type a `?` now. After it finishes, type a `SPC`.

(If you are using the standalone Info reader, type 'l' to return here.)

From now on, you will encounter large nodes without warning, and will be expected to know how to use Space and Delete to move around in them without being told. Since not all terminals have the same size screen, it would be impossible to warn you anyway.

>> Now type `n` to see the description of the `m` command.

1.5 Menus

Menus and the `m` command

With only the `n` and `p` commands for moving between nodes, nodes are restricted to a linear sequence. Menus allow a branching structure. A menu is a list of other nodes you can move to. It is actually just part of the text of the node formatted specially so that Info can interpret it. The beginning of a menu is always identified by a line which starts with '`* Menu:`'. A node contains a menu if and only if it has a line in it which starts that way. The only menu you can use at any moment is the one in the node you are in. To use a menu in any other node, you must move to that node first.

After the start of the menu, each line that starts with a '`*`' identifies one subtopic. The line usually contains a brief name for the subtopic (followed by a '`:`'), the name of the node that talks about that subtopic, and optionally some further description of the subtopic. Lines in the menu that do not start with a '`*`' have no special meaning—they are only for the human reader's benefit and do not define additional subtopics. Here is an example:

```
* Foo:  FOO's Node      This tells about FOO
```

The subtopic name is Foo, and the node describing it is '`FOO's Node`'. The rest of the line is just for the reader's Information. [[But this line is not a real menu item, simply because there is no line above it which starts with '`* Menu:`'.]]

When you use a menu to go to another node (in a way that will be described soon), what you specify is the subtopic name, the first thing in the menu line. Info uses it to find the menu line, extracts the node name from it, and goes to that node. The reason that there is both a subtopic name and a node name is that the node name must be meaningful to the computer and may therefore have to be ugly looking. The subtopic name can be chosen just to be convenient for the user to specify. Often the node name is convenient for the user to specify and so both it and the subtopic name are the same. There is an abbreviation for this:

```
* Foo::   This tells about FOO
```

This means that the subtopic name and node name are the same; they are both '`Foo`'.

>> Now use Spaces to find the menu in this node, then come back to the front with a `b` and some Spaces. As you see, a menu is actually visible in its node. If you cannot find a menu in a node by looking at it, then the node does not have a menu and the `m` command is not available.

The command to go to one of the subnodes is `m`—but *do not do it yet!* Before you use `m`, you must understand the difference between commands and arguments. So far, you have learned several commands that do not need arguments. When you type one, Info processes it and is instantly ready for another command. The `m` command is different: it is incomplete without the *name of the subtopic*. Once you have typed `m`, Info tries to read the subtopic name.

Now look for the line containing many dashes near the bottom of the screen. There is one more line beneath that one, but usually it is blank. If it is empty, Info is ready for a command, such as `n` or `b` or Space or `m`. If that line contains text ending in a colon, it mean Info is trying to read the *argument* to a command. At such times, commands do not work, because Info tries to use them as the argument. You must either type the argument and finish the command you started, or type `Control-g` to cancel the command. When you have done one of those things, the line becomes blank again.

The command to go to a subnode via a menu is `m`. After you type the `m`, the line at the bottom of the screen says '`Menu item: `'. You must then type the name of the subtopic you want, and end it with a `RET`.

You can abbreviate the subtopic name. If the abbreviation is not unique, the first matching subtopic is chosen. Some menus put the shortest possible abbreviation for each subtopic name in capital letters, so you can see how much you need to type. It does not matter whether you use upper case or lower case when you type the subtopic. You should not put any spaces at the end, or inside of the item name, except for one space where a space appears in the item in the menu.

You can also use the *completion* feature to help enter the subtopic name. If you type the Tab key after entering part of a name, it will magically fill in more of the name—as much as follows uniquely from what you have entered.

If you move the cursor to one of the menu subtopic lines, then you do not need to type the argument: you just type a Return, and it stands for the subtopic of the line you are on.

Here is a menu to give you a chance to practice.

* Menu: The menu starts here.

This menu gives you three ways of going to one place, Help-FOO.

* Foo: Help-FOO. A node you can visit for fun.
* Bar: Help-FOO. Strange! two ways to get to the same place.
* Help-FOO:: And yet another!

>> Now type just an `m` and see what happens:

Now you are "inside" an `m` command. Commands cannot be used now; the next thing you will type must be the name of a subtopic.

You can change your mind about doing the `m` by typing Control-g.

>> Try that now; notice the bottom line clear.

>> Then type another `m`.

>> Now type '`BAR`' item name. Do not type Return yet.

While you are typing the item name, you can use the Delete key to cancel one character at a time if you make a mistake.

>> Type one to cancel the '`R`'. You could type another '`R`' to replace it. You do not have to, since '`BA`' is a valid abbreviation.

>> Now you are ready to go. Type a `RET`.

After visiting Help-FOO, you should return here.

>> Type `n` to see more commands.

Here is another way to get to Help-FOO, a menu. You can ignore this if you want, or else try it (but then please come back to here).

1.5.1 The `u` command

Congratulations! This is the node '`Help-FOO`'. Unlike the other nodes you have seen, this one has an '`Up`': '`Help-M`', the node you just came from via the `m` command. This is the usual convention—the nodes you reach from a menu have '`Up`' nodes that lead back to the menu. Menus move Down in the tree, and '`Up`' moves Up. '`Previous`', on the other hand, is usually used to "stay on the same level but go backwards"

You can go back to the node '`Help-M`' by typing the command `u` for "Up". That puts you at the *front* of the node—to get back to where you were reading you have to type some `SPC`s.

>> Now type `u` to move back up to '`Help-M`'.

1.6 Some advanced Info commands

The course is almost over, so please stick with it to the end.

If you have been moving around to different nodes and wish to retrace your steps, the `l` command (`l` for *last*) will do that, one node-step at a time. As you move from node to node, Info records the nodes where you have been in a special history list. The `l` command revisits nodes in the history list; each successive `l` command moves one step back through the history.

If you have been following directions, ad `l` command now will get you back to '`Help-M`'. Another `l` command would undo the `u` and get you back to '`Help-FOO`'. Another `l` would undo the `m` and get you back to '`Help-M`'.

>> Try typing three `l`'s, pausing in between to see what each `l` does.

Then follow directions again and you will end up back here.

Note the difference between `l` and `p`: `l` moves to where *you* last were, whereas `p` always moves to the node which the header says is the '`Previous`' node (from this node, to '`Help-M`').

The '`d`' command gets you instantly to the Directory node. This node, which is the first one you saw when you entered Info, has a menu which leads (directly, or indirectly through other menus), to all the nodes that exist.

>> Try doing a '`d`', then do an `l` to return here (yes, *do* return).

Sometimes, in Info documentation, you will see a cross reference. Cross references look like this: See [Help-Cross], page 8. That is a real, live cross reference which is named '`Cross`' and points at the node named '`Help-Cross`'.

If you wish to follow a cross reference, you must use the '`f`' command. The '`f`' must be followed by the cross reference name (in this case, '`Cross`'). While you enter the name, you can use the Delete key to edit your input. If you change your mind about following any reference, you can use `Control-g` to cancel the command.

Completion is available in the '`f`' command; you can complete among all the cross reference names in the current node by typing a Tab.

>> Type '`f`', followed by '`Cross`', and a `RET`.

To get a list of all the cross references in the current node, you can type `?` after an '`f`'. The '`f`' continues to await a cross reference name even after printing the list, so if you don't actually want to follow a reference, you should type a `Control-g` to cancel the '`f`'.

>> Type "f?" to get a list of the cross references in this node. Then type a `Control-g` and see how the '`f`' gives up.

>> Now type `n` to see the last node of the course.

The node reached by the cross reference in Info

This is the node reached by the cross reference named '`Cross`'.

While this node is specifically intended to be reached by a cross reference, most cross references lead to nodes that "belong" someplace else far away in the structure of Info. So you cannot expect the footnote to have a 'Next', 'Previous' or 'Up' pointing back to where you came from. In general, the `l` (el) command is the only way to get back there.

>> Type `l` to return to the node where the cross reference was.

1.7 Quitting Info

To get out of Info, back to what you were doing before, type `q` for *Quit*.

This is the end of the course on using Info. There are some other commands that are meant for experienced users; they are useful, and you can find them by looking in the directory node for documentation on Info. Finding them will be a good exercise in using Info in the usual manner.

>> Type '`d`' to go to the Info directory node; then type '`mInfo`' and Return, to get to the node about Info and see what other help is available.

2 Info for Experts

This chapter describes various advanced Info commands, and how to write an Info as distinct from a Texinfo file. (However, in most cases, writing a Texinfo file is better, since you can use it *both* to generate an Info file and to make a printed manual. See section "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*.)

2.1 Advanced Info Commands

`g`, `s`, `1`, – `9`, and `e`

If you know a node's name, you can go there by typing `g`, the name, and `RET`. Thus, `gTopRET` would go to the node called '`Top`' in this file (its directory node). `gExpertRET` would come back here.

Unlike `m`, `g` does not allow the use of abbreviations.

To go to a node in another file, you can include the filename in the node name by putting it at the front, in parentheses. Thus, `g(dir)TopRET` would go to the Info Directory node, which is node '`Top`' in the file '`dir`'.

The node name '`*`' specifies the whole file. So you can look at all of the current file by typing `g*RET` or all of any other file with `g(FILENAME)RET`.

The `s` command allows you to search a whole file for a string. It switches to the next node if and when that is necessary. You type `s` followed by the string to search for, terminated by `RET`. To search for the same string again, just `s` followed by `RET` will do. The file's nodes are scanned in the order they are in in the file, which has no necessary relationship to the order that they may be in in the tree structure of menus and '`next`' pointers. But normally the two orders are not very different. In any case, you can always do a `b` to find out what node you have reached, if the header is not visible (this can happen, because `s` puts your cursor at the occurrence of the string, not at the beginning of the node).

If you grudge the system each character of type-in it requires, you might like to use the commands `1`, `2`, `3`, `4`, ... `9`. They are short for the `m` command together with an argument. `1` goes through the first item in the current node's menu; `2` goes through the second item, etc.

If you display supports multiple fonts, and you are using Emacs' Info mode to read Info files, the '`*`' for the fifth menu item is underlines, and so is the '`*`' for the ninth item; these underlines make it easy to see at a glance which number to use for an item.

On ordinary terminals, you won't have underlining. If you need to actually count items, it is better to use `m` instead, and specify the name.

The Info command `e` changes from Info mode to an ordinary Emacs editing mode, so that you can edit the text of the current node. Type `C-c C-c` to switch back to Info. The `e` command is allowed only if the variable `Info-enable-edit` is non-`nil`.

2.2 Adding a new node to Info

To add a new topic to the list in the Info directory, you must:

1. Create some nodes, in some file, to document that topic.
2. Put that topic in the menu in the directory. See Section 2.3 [Menus], page 13.

Usually, the way to create the nodes is with Texinfo see section "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*); this has the advantage that you can also make a printed manual from them. However, if hyou want to edit an Info file, here is how.

The new node can live in an existing documentation file, or in a new one. It must have a `^_` character before it (invisible to the user; this node has one but you cannot see it), and it ends with either a `^_`, a `^L`, or the end of file. Note: If you put in a `^L` to end a new node, be sure that there is a `^_` after it to start the next one, since `^L` cannot *start* a node. Also, a nicer way to make a node boundary be a page boundary as well is to put a `^L` *right after* the `^_`.

The `^_` starting a node must be followed by a newline or a `^L` newline, after which comes the node's header line. The header line must give the node's name (by which Info finds it), and state the names of the '`Next`', '`Previous`', and '`Up`' nodes (if there are any). As you can see, this node's '`Up`' node is the node '`Top`', which points at all the documentation for Info. The '`Next`' node is '`Menus`'.

The keywords *Node*, *Previous*, *Up*, and *Next*, may appear in any order, anywhere in the header line, but the recommended order is the one in this sentence. Each keyword must be followed by a colon, spaces and tabs, and then the appropriate name. The name may be terminated with a tab, a comma, or a newline. A space does not end it; node names may contain spaces. The case of letters in the names is insignificant.

A node name has two forms. A node in the current file is named by what appears after the '`Node: `' in that node's first line. For example, this node's name is '`Add`'. A node in another file is named by '`(`*filename*`)`*node-within-file*', as in '`(info)Add`' for this node. If the file name starts with "./", then it is relative to the current directory; otherwise, it is relative starting from the standard Info file directory of your site. The name '`(`*filename*`)Top`' can be abbreviated to just '`(`*filename*`)`'. By convention, the name '`Top`' is used for the "highest" node in any single file—the node whose '`Up`' points out of the file. The Directory node is '`(dir)`'. The '`Top`' node of a document file listed in the Directory should have an '`Up: (dir)`' in it.

The node name `*` is special: it refers to the entire file. Thus, `g*` shows you the whole current file. The use of the node `*` is to make it possible to make old-fashioned, unstructured files into nodes of the tree.

The '`Node:`' name, in which a node states its own name, must not contain a filename, since Info when searching for a node does not expect one to be there. The '`Next`', '`Previous`' and '`Up`' names may contain them. In this node, since the '`Up`' node is in the same file, it was not necessary to use one.

Note that the nodes in this file have a file name in the header line. The file names are ignored by Info, but they serve as comments to help identify the node for the user.

2.3 How to Create Menus

Any node in the Info hierarchy may have a *menu*—a list of subnodes. The `m` command searches the current node's menu for the topic which it reads from the terminal.

A menu begins with a line starting with '`* Menu:`'. The rest of the line is a comment. After the starting line, every line that begins with a '`* `' lists a single topic. The name of the topic–the argument that the user must give to the `m` command to select this topic—comes right after the star and space, and is followed by a colon, spaces and tabs, and the name of the node which discusses that topic. The node name, like node names following '`Next`', '`Previous`' and '`Up`', may be terminated with a tab, comma, or newline; it may also be terminated with a period.

If the node name and topic name are the same, then rather than giving the name twice, the abbreviation '`* NAME::`' may be used (and should be used, whenever possible, as it reduces the visual clutter in the menu).

It is considerate to choose the topic names so that they differ from each other very near the beginning—this allows the user to type short abbreviations. In a long menu, it is a good idea to capitalize the beginning of each item name which is the minimum acceptable abbreviation for it (a long menu is more than 5 or so entries).

The nodes listed in a node's menu are called its "subnodes", and it is their "superior". They should each have an '`Up:`' pointing at the superior. It is often useful to arrange all or most of the subnodes in a sequence of '`Next`' and '`Previous`' pointers so that someone who wants to see them all need not keep revisiting the Menu.

The Info Directory is simply the menu of the node '`(dir)Top`'—that is, node '`Top`' in file '`.../info/dir`'. You can put new entries in that menu just like any other menu. The Info Directory is *not* the same as the file directory called '`info`'. It happens that many of Info's files live on that file directory, but they do not have to; and files on that directory are not automatically listed in the Info Directory node.

Also, although the Info node graph is claimed to be a "hierarchy", in fact it can be *any* directed graph. Shared structures and pointer cycles are perfectly possible, and can be used if they are appropriate to the meaning to be expressed. There is no need for all the nodes in a file to form a connected structure. In fact, this file has two connected components. You are in one of them, which is under the node '`Top`'; the other contains the node '`Help`' which the `h` command goes to. In fact, since there is no garbage collector, nothing terrible happens if a substructure is not pointed to, but such a substructure is rather useless since nobody can ever find out that it exists.

2.4 Creating Cross References

A cross reference can be placed anywhere in the text, unlike a menu item which must go at the front of a line. A cross reference looks like a menu item except that it has '`*note`' instead of `*`. It *cannot* be terminated by a '`)`', because '`)`''s are so often part of node names. If you wish to enclose a cross reference in parentheses, terminate it with a period first. Here are two examples of cross references pointers:

```
*Note details: commands.  (See *note 3: Full Proof.)
```

They are just examples. The places they "lead to" do not really exist!

2.5 Tag Tables for Info Files

You can speed up the access to nodes of a large Info file by giving it a tag table. Unlike the tag table for a program, the tag table for an Info file lives inside the file itself and is used automatically whenever Info reads in the file.

To make a tag table, go to a node in the file using Emacs Info mode and type `M-x Info-tagify`. Then you must use `C-x C-s` to save the file.

Once the Info file has a tag table, you must make certain it is up to date. If, as a result of deletion of text, any node moves back more than a thousand characters in the file from the position recorded in the tag table, Info will no longer be able to find that node. To update the tag table, use the `Info-tagify` command again.

An Info file tag table appears at the end of the file and looks like this:

```
^_

Tag Table:
File: info, Node: Cross-refs^?21419
File: info,  Node: Tags^?22145
^_
End Tag Table
```

Note that it contains one line per node, and this line contains the beginning of the node's header (ending just after the node name), a Delete character, and the character position in the file of the beginning of the node.

2.6 Checking an Info File

When creating an Info file, it is easy to forget the name of a node when you are making a pointer to it from another node. If you put in the wrong name for a node, this is not detected until someone tries to go through the pointer using Info. Verification of the Info file is an automatic process which checks all pointers to nodes and reports any pointers which are invalid. Every '`Next`', '`Previous`', and '`Up`' is checked, as is every menu item and every cross reference. In addition, any '`Next`' which does not have a ‘Previous’ pointing back is reported. Only pointers within the file are checked, because checking pointers to other files would be terribly slow. But those are usually few.

To check an Info file, do `M-x Info-validate` while looking at any node of the file with Emacs Info mode.

2.7 Emacs Info-mode Variables

The following variables may modify the behaviour of Info-mode in Emacs; you may wish to set one or several of these variables interactively, or in your ‘`~/.emacs`’ init file. See section “Examining and Setting Variables” in *The GNU Emacs Manual*.

`Info-enable-edit`
: Set to `nil`, disables the ‘`e`’ (`Info-edit`) command. A non-`nil` value enables it. See Section 2.2 [Add], page 12.

`Info-enable-active-nodes`
: When set to a non-`nil` value, allows Info to execute Lisp code associated with nodes. The Lisp code is executed when the node is selected.

`Info-directory-list`
: The list of directories to search for Info files. Each element is a string (directory name) or `nil` (try default directory).

`Info-directory`
: The standard directory for Info documentation files. Only used when the function `Info-directory` is called.

3 Creating an Info File from a Makeinfo file

`makeinfo` is a utility that converts a Texinfo file into an Info file; `texinfo-format-region` and `texinfo-format-buffer` are GNU Emacs functions that do the same.

See section "Creating an Info File" in *the Texinfo Manual*, to learn how to create an Info file from a Texinfo file.

See section "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*, to learn how to write a Texinfo file.