Info

The online, hyper-text GNU documentation system

Brian Fox
and the GNU Texinfo community

This file describes how to use Info, the on-line, menu-driven GNU documentation system.

Table of Contents

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 briefly explains how to generate Info files from Texinfo files.

This manual is primarily designed for browsing with an Info reader program on a computer, so that you can try Info commands while reading about them. Reading it on paper or with an HTML browser 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 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 a relatively small number of lines on its screen, it is necessary to give you special advice at the beginning.

If you see the text '`--All----`' 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 ⟨SPC⟩, the Space bar. To move back up, press the key labeled '`Backspace`' or '`DEL`' (on some keyboards, this key might be labeled '`Delete`').

1.2 How to use Info

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

There are two ways to use Info: from within Emacs or as a stand-alone reader that you can invoke from a shell using the command `info`.

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 mode line says that this is node '`Help`' in the file '`info`'.

The top line of a node is its *header*. This node's header (look at it now) 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. In the stand-alone Info reader program, the header line shows the names of this node and the info file as well. In Emacs, the header line is

duplicated in a special typeface, and the duplicate remains at the top of the window all the time even if you scroll through the node.

Besides a '`Next`', a node can have a '`Previous`' or an '`Up`' links, or both. As you can see, this node has all of these links.

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.

>> If you are in Emacs and have a mouse, and if you already practiced
typing *n* to get to the next node, click now with the middle
mouse button on the '`Next`' link to do the same "the mouse way".

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 type *n* yet. First, try the *p* command,
or click the middle mouse button on the '`Prev`' link. That
takes you to the '`Previous`' node. Then use *n* to return here.

If you read this in Emacs, you will see an '`Info`' item in the menu bar, close to its right edge. Clicking the mouse on the '`Info`' menu-bar item opens a menu of commands which include '`Next`' and '`Prev`' (and also some others which you didn't yet learn about).

This all probably seems insultingly simple so far, but *please don't* start skimming. Things will get complicated soon enough! Also, please do not try a new command until you are told it is time to. You could make Info skip past an important warning that was coming up.

>> Now do an *n*, or click the middle mouse button on the '`Next`'
link, to get to the node '`Help-^L`' and learn more.

1.4 The Space, DEL, B and ^L commands

This node's mode line tells you that you are now at node '`Help-^L`', and the header line tells you that *p* would get you back to '`Help-P`'. The node's title is highlighted and may be underlined as well; it says what the node is about.

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 ⟨SPC⟩, ⟨BACKSPACE⟩ (or ⟨DEL⟩)[1] and *b* commands exist to allow you to "move around" in a node that does not all fit on the screen at once. ⟨SPC⟩ moves forward, to show what was below the bottom of the screen. ⟨DEL⟩ or ⟨BACKSPACE⟩ moves backward, to show what

[1] The key which we call "Backspace or DEL" in this manual is labeled differently on different keyboards. Look for a key which is a little ways above the ⟨ENTER⟩ or ⟨RET⟩ key and which you normally use outside Emacs to erase the character before the cursor, i.e. the character you typed last. It might be labeled '`Backspace`' or '`<-`' or '`DEL`', or sometimes '`Delete`'.

was above the top of the screen (there is not anything above the top until you have typed some spaces).

>> Now try typing a ⟨SPC⟩ (afterward, type a ⟨BACKSPACE⟩ to return here).

When you type the ⟨SPC⟩, the two lines that were at the bottom of the screen appear at the top, followed by more lines. ⟨DEL⟩ or ⟨BACKSPACE⟩ 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 are reading this in Emacs, note that the header line is always visible, never scrolling off the display. That way, you can always see the '`Next`', '`Prev`', and '`Up`' links, and you can conveniently go to one of these links at any time by clicking the middle mouse button on the link.

⟨SPC⟩ and ⟨DEL⟩ not only move forward and backward through the current node. They also move between nodes. ⟨SPC⟩ at the end of a node moves to the next node; ⟨DEL⟩ (or ⟨BACKSPACE⟩) at the beginning of a node moves to the previous node. In effect, these commands scroll through all the nodes in an Info file as a single logical sequence. You can read an entire manual top to bottom by just typing ⟨SPC⟩, and move backward through the entire manual from bottom to top by typing ⟨DEL⟩ (or ⟨BACKSPACE⟩).

In this sequence, a node's subnodes appear following their parent. If a node has a menu, ⟨SPC⟩ takes you into the subnodes listed in the menu, one by one. Once you reach the end of a node, and have seen all of its subnodes, ⟨SPC⟩ takes you to the next node or to the parent's next node.

Many keyboards nowadays have two scroll keys labeled '`PageUp`' and '`PageDown`' (or maybe '`Prior`' and '`Next`'). If your keyboard has these keys, you can use them to move forward and backward through the text of one node, like ⟨SPC⟩ and ⟨BACKSPACE⟩ (or ⟨DEL⟩). However, ⟨PAGEUP⟩ and ⟨PAGEDOWN⟩ keys never scroll beyond the beginning or the end of the current node.

If your screen is ever garbaged, you can tell Info to display it again by typing *`C-l`* (*`Control-L`*, that is—hold down ⟨CTRL⟩ and type *`L`* or *`l`*).

>> Type *`C-l`* now.

To move back to the beginning of the node you are on, you can type the ⟨BACKSPACE⟩ key (or ⟨DEL⟩) many times. You can also type *`b`* just once. *`b`* stands 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, by typing ⟨SPC⟩ one or more times.

If your screen is very tall, all of this node might fit at once. In that case, *`b`* won't do anything. But you could observe the effect of the *`b`* key if you use a smaller window.

You have just learned a considerable number of commands. If you want to use one but have trouble remembering which, you should type a *`?`* (in Emacs it runs the `Info-summary` command) which displays a brief list of commands. When you are finished looking at the list, make it go away by typing a ⟨SPC⟩ repeatedly.

>> Type a ⟨?⟩ now. Press ⟨SPC⟩ to see consecutive screenfuls of the list until finished. Then type ⟨SPC⟩ several times. If

you are using Emacs, the help will then go away automatically.

(If you are using the stand-alone Info reader, type *C-x 0* to return here, that is—press and hold ⟨CTRL⟩, type an *x*, then release ⟨CTRL⟩ and *x*, and press *0*—a zero, not the letter "o".)

From now on, you will encounter large nodes without warning, and will be expected to know how to use ⟨SPC⟩ and ⟨BACKSPACE⟩ 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*, or click the middle mouse button on the '`Next`' link,
to visit the next node.

1.5 Invisible text in Emacs Info

Before discussing menus, we need to make some remarks that are only relevant to users reading Info using Emacs. Users of the stand-alone version can skip this node by typing *]* now.

In Emacs, certain text that appears in the stand-alone version is normally hidden, technically because it has the '`invisibility`' property. Invisible text is really a part of the text. It becomes visible (by default) after killing and yanking, it appears in printed output, it gets saved to file just like any other text, and so on. Thus it is useful to know it is there.

You can make invisible text visible by using the command *M-x visible-mode*. Visible mode is a minor mode, so using the command a second time will make the text invisible again. Watch the effects of the command on the "menu" below and the top line of this node.

If you prefer to *always* see the invisible text, you can set `Info-hide-note-references` to `nil`. Enabling Visible mode permanently is not a real alternative, because Emacs Info also uses (although less extensively) another text property that can change the text being displayed, the '`display`' property. Only the invisibility property is affected by Visible mode. When, in this tutorial, we refer to the '`Emacs`' behavior, we mean the *default* Emacs behavior.

Now type *]*, to learn about the *]* and *[* commands.

1.5.1 The *]* and *[* commands

If you type *n* now, you get an error message saying that this node has no next node. Similarly, if you type *p*, the error message tells you that there is no previous node. (The exact message depends on the Info reader you use.) This is because *n* and *p* carry you to the next and previous node *at the same level*. The present node is contained in a menu (see next) of the node you came from, and hence is considered to be at a lower level. It is the only node in the previous node's menu (even though it was listed three times). Hence it has no next or previous node that *n* or *p* could move to.

If you systematically move through a manual by typing *n*, you run the risk of skipping many nodes. You do not run this risk if you systematically use ⟨SPC⟩, because, when you scroll to the bottom of a node and type another ⟨SPC⟩, then this carries you to the following node in the manual *regardless of level*. If you immediately want to go to that node, without having to scroll to the bottom of the screen first, you can type *]*.

Similarly, ⟨BACKSPACE⟩ carries you to the preceding node regardless of level, after you scrolled to the beginning of the present node. If you want to go to the preceding node immediately, you can type *[*.

For instance, typing this sequence will come back here in three steps: *[n [*. To do the same backward, type *] p]*.

Now type *]* to go to the next node and learn about menus.

1.6 Menus and the *m* command

With only the *n* (next), *p* (previous), ⟨SPC⟩, ⟨BACKSPACE⟩, *]* and *[* 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 '`:`', normally hidden in Emacs), the name of the node that talks about that subtopic (again, normally hidden in Emacs), 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:  Node about FOO.      This tells about FOO.
```

The subtopic name is Foo, and the node describing it is '`Node about FOO`'. 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:`'. Also, in a real menu item, the '`*`' would appear at the very start of the line. This is why the "normally hidden" text in Emacs, namely '`: Node about FOO.`', is actually visible in this example, even when Visible mode is off.]]

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`'. (The '`::`' is normally hidden in Emacs.)

>> Now use ⟨SPC⟩ to find the menu in this node, then come back to the front with a *b* and some ⟨SPC⟩s. 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.

If you keep typing ⟨SPC⟩ once the menu appears on the screen, it will move to another node (the first one in the menu). If that happens, type ⟨BACKSPACE⟩ to come back.

The command to go to one of the subnodes is *m*. This is very different from the commands you have used: it is a command that prompts you for more input.

The Info commands you know do not need additional input; when you type one of them, Info processes it instantly and then is ready for another command. The *m* command is different: it needs to know the *name of the subtopic*. Once you have typed *m*, Info tries to read the subtopic name.

Now, in the stand-alone Info, look for the line containing many dashes near the bottom of the screen. (This is the stand-alone equivalent for the mode line in Emacs.) There is one more line beneath that one, but usually it is blank. (In Emacs, this is the echo area.) When it is blank, Info is ready for a command, such as *n* or *b* or ⟨SPC⟩ or *m*. If that line contains text ending in a colon, it means Info is reading more input for the last command. You can't type an Info command then, because Info is trying to read input, not commands. You must either give the input and finish the command you started, or type *Control-g* to cancel the command. When you have done one of those things, the input entry line becomes blank again. Then you can type Info commands 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⟩. In Emacs, *m* runs the command `Info-menu`.

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 fill in more of the name—as much as Info can deduce from the part 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 ⟨RET⟩, and it stands for the subtopic of the line you are on. You can also click the middle mouse button directly on the subtopic line to go there.

Here is a menu to give you a chance to practice. This menu gives you three ways of going to one place, Help-FOO:

(Turn Visible mode on if you are using Emacs.)

>> 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*, the item name. Do not type ⟨RET⟩ yet.

While you are typing the item name, you can use the ⟨DEL⟩ (or ⟨BACKSPACE⟩) key to cancel one character at a time if you make a mistake.

>> Press ⟨DEL⟩ to cancel the 'R'. You could type another *R* to replace it. But 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.

Another way to move to the menu subtopic lines and between them is to type ⟨TAB⟩. Each time you type a ⟨TAB⟩, you move to the next subtopic line. To move to a previous subtopic line, type *M-*⟨TAB⟩—that is, press and hold the ⟨META⟩ key and then press ⟨TAB⟩. (On some keyboards, the ⟨META⟩ key might be labeled 'Alt'.)

Once you move cursor to a subtopic line, press ⟨RET⟩ to go to that subtopic's node.

If your terminal supports a mouse, you have yet another way of going to a subtopic. Move your mouse pointer to the subtopic line, somewhere between the beginning '*' and the colon ':' which ends the subtopic's brief name. You will see the subtopic's name change its appearance (usually, its background color will change), and the shape of the mouse pointer will change if your platform supports that. After a while, if you leave the mouse on that spot, a small window will pop up, saying "Mouse-2: go to that node", or the same message may appear at the bottom of the screen.

Mouse-2 is the second button of your mouse counting from the left—the middle button on a 3-button mouse. (On a 2-button mouse, you may have to press both buttons together to "press the middle button".) The message tells you pressing *Mouse-2* with the current position of the mouse pointer (on subtopic in the menu) will go to that subtopic.

More generally, *Mouse-2* in an Info buffer finds the nearest link to another node and goes there. For example, near a cross reference it acts like *f*, in a menu it acts like *m*, on the node's header line it acts like *n*, *p*, or *u*, etc. At end of the node's text *Mouse-2* moves to the next node, or up if there's no next node.

>> Type *n* to see more commands.

1.6.1 The *u* command

Congratulations! This is the node 'Help-FOO'. It has an 'Up' pointer '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" (the Emacs command run by *u* is Info-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. (Some Info readers, such as the one built into Emacs, put you at the same place where you were reading in 'Help-M'.)

Another way to go Up is to click *Mouse-2* on the 'Up' pointer shown in the header line (provided that you have a mouse).

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

1.7 Following Cross-References

In Info documentation, you will see many *cross references*. Cross references look like this: See ⟨undefined⟩ [Help-Cross], page ⟨undefined⟩. That text is a real, live cross reference, whose name is '`Cross`' and which points to the node named '`Help-Cross`'. (The node name is hidden in Emacs. Do `M-x visible-mode` to show or hide it.)

There are two ways to follow a cross reference. You can move the cursor to it and press ⟨RET⟩, just as in a menu. ⟨RET⟩ follows the cross reference that the cursor is on. Or you can type `f` and then specify the name of the cross reference (in this case, '`Cross`') as an argument. In Emacs Info, `f` runs `Info-follow-reference`,

In the `f` command, you select the cross reference with its name, so it does not matter where the cursor was. If the cursor is on or near a cross reference, `f` suggests that reference name in parentheses as the default; typing ⟨RET⟩ will follow that reference. However, if you type a different reference name, `f` will follow the other reference which has that name.

>> Type `f`, followed by `Cross`, and then ⟨RET⟩.

As you enter the reference name, you can use the ⟨DEL⟩ (or ⟨BACKSPACE⟩) 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⟩.

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 displaying 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.

The ⟨TAB⟩ and `M-`⟨TAB⟩ key, which move between menu items in a menu, also move between cross references outside of menus.

Sometimes a cross reference (or a node) can lead to another file (in other words another "manual"), or, on occasion, even a file on a remote machine (although Info files distributed with Emacs or the stand-alone Info avoid using remote links). Such a cross reference looks like this: See section "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*. (After following this link, type `l` to get back to this node.) Here the name '`texinfo`' between parentheses (shown in the stand-alone version) refers to the file name. This file name appears in cross references and node names if it differs from the current file. In Emacs, the file name is hidden (along with other text). (Use `M-x visible-mode` to show or hide it.)

The remainder of this node applies only to the Emacs version. If you use the stand-alone version, you can type `n` immediately.

To some users, switching manuals is a much bigger switch than switching sections. These users like to know that they are going to be switching to another manual (and which one) before actually doing so, especially given that, if one does not notice, Info commands like `t` (see the next node) can have confusing results.

If you put your mouse over the cross reference and if the cross reference leads to a different manual, then the information appearing in a separate box (tool tip) or in the echo area, will mention the file the cross reference will carry you to (between parentheses). This

is also true for menu subtopic names. If you have a mouse, just leave it over the '`Overview`' cross reference above and watch what happens.

If you always like to have that information available without having to move your mouse over the cross reference, set `Info-hide-note-references` to a value other than t (see ⟨undefined⟩ [Emacs Info Variables], page ⟨undefined⟩). You might also want to do that if you have a lot of cross references to files on remote machines and have non-permanent or slow access, since otherwise you might not be able to distinguish between local and remote links.

>> Now type `n` to learn more commands.

1.8 Some intermediate Info commands

The introductory course is almost over; please continue a little longer to learn some intermediate-level commands.

Most Info files have an index, which is actually a large node containing little but a menu. The menu has one menu item for each topic listed in the index. (As a special feature, menus for indices may also include the line number within the node of the index entry. This allows Info readers to go to the exact line of an entry, not just the start of the containing node.)

You can get to the index from the main menu of the file with the `m` command; then you can use the `m` command again in the index node to go to the node that describes the topic you want.

There is also a short-cut Info command, `i`, which does all of that for you. It searches the index for a given topic (a string) and goes to the node which is listed in the index for that topic. See ⟨undefined⟩ [Info Search], page ⟨undefined⟩, for a full explanation.

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.

In Emacs, `l` runs the command `Info-last`.

>> Try typing `p p n` and then three `l`'s, pausing in between
to see what each `l` does. You should wind up right 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, the '`Prev`' link leads to '`Help-Xref`').

The `d` command (`Info-directory` in Emacs) 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. The Directory node lists all the manuals and other Info documents that are, or could be, installed on your system.

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

The `t` command moves to the '`Top`' node of the manual. This is useful if you want to browse the manual's main menu, or select some specific top-level menu item. The Emacs command run by `t` is `Info-top-node`.

Clicking *Mouse-2* on or near a cross reference also follows the reference. You can see that the cross reference is mouse-sensitive by moving the mouse pointer to the reference and watching how the underlying text and the mouse pointer change in response.

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

See ⟨undefined⟩ [Expert Info], page ⟨undefined⟩, for more advanced Info features.

2 Info for Experts

This chapter describes various Info commands for experts. (If you are using a stand-alone Info reader, there are additional commands specific to it, which are documented in several chapters of section "GNU Info" in *GNU Info*.)

This chapter also explains 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 to make a printed manual or produce other formats, such as HTML and DocBook, as well as for generating Info files.) See section "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*.

2.1 Advanced Info Commands

Here are some more Info commands that make it easier to move around.

g goes to a node by name

If you know a node's name, you can go there by typing *g*, the name, and ⟨RET⟩. Thus, *gTop*⟨RET⟩ would go to the node called '`Top`' in this file. (This is equivalent to *t*, see ⟨undefined⟩ [Help-Int], page ⟨undefined⟩.) *gAdvanced*⟨RET⟩ would come back here. *g* in Emacs runs the command `Info-goto-node`.

Unlike *m*, *g* does not allow the use of abbreviations. But it does allow completion, so you can type ⟨TAB⟩ to complete a partial node name.

To go to a node in another file, you can include the file name in the node name by putting it at the front, in parentheses. Thus, *g(dir)Top*⟨RET⟩ would go to the Info Directory node, which is the node '`Top`' in the Info file '`dir`'. Likewise, *g(emacs)Top*⟨RET⟩ goes to the top node of the Emacs manual.

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⟩.

1 – *9* choose a menu subtopic by its number

If you begrudge each character of type-in which your system requires, you might like to use the commands *1*, *2*, *3*, *4*, ..., *9*. They are short for the *m* command together with a name of a menu subtopic. *1* goes through the first item in the current node's menu; *2* goes through the second item, etc. In the stand-alone reader, *0* goes through the last menu item; this is so you need not count how many entries are there. In Emacs, the digit keys run the command `Info-nth-menu-item`.

If your display supports multiple fonts, colors or underlining, and you are using Emacs' Info mode to read Info files, the third, sixth and ninth menu items have a '`*`' that stands out, either in color or in some other attribute, such as underline; this makes it easy to see at a glance which number to use for an item.

Some terminals don't support either multiple fonts, colors or underlining. If you need to actually count items, it is better to use *m* instead, and specify the name, or use ⟨TAB⟩ to quickly move between menu items.

`e` makes Info document editable

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`.

The `e` command only works in Emacs, where it runs the command `Info-edit`. The stand-alone Info reader doesn't allow you to edit the Info file, so typing `e` there goes to the end of the current node.

`M-n` creates a new independent Info buffer in Emacs

If you are reading Info in Emacs, you can select a new independent Info buffer in another window by typing `M-n`. The new buffer starts out as an exact copy of the old one, but you will be able to move independently between nodes in the two buffers. (In Info mode, `M-n` runs the Emacs command `clone-buffer`.)

In Emacs Info, you can also produce new Info buffers by giving a numeric prefix argument to the `m` and `g` commands. `C-u m` and `C-u g` go to a new node in exactly the same way that `m` and `g` do, but they do so in a new Info buffer which they select in another window.

2.2 How to search Info documents for specific subjects

The commands which move between and inside nodes allow you to read the entire manual or its large portions. But what if you need to find some information in the manual as fast as you can, and you don't know or don't remember in what node to look for it? This need arises when you use a manual as a *reference*, or when it is impractical to read the entire manual before you start using the programs it describes.

Info has powerful searching facilities that let you find things quickly. You can search either the manual indices or its text.

Since most subjects related to what the manual describes should be indexed, you should try the index search first. The `i` command prompts you for a subject and then looks up that subject in the indices. If it finds an index entry with the subject you typed, it goes to the node to which that index entry points. You should browse through that node to see whether the issue you are looking for is described there. If it isn't, type `,` one or more times to go through additional index entries which match your subject.

The `i` command finds all index entries which include the string you typed *as a substring*. For each match, Info shows in the echo area the full index entry it found. Often, the text of the full index entry already gives you enough information to decide whether it is relevant to what you are looking for, so we recommend that you read what Emacs shows in the echo area before looking at the node it displays.

Since `i` looks for a substring, you can search for subjects even if you are not sure how they are spelled in the index. For example, suppose you want to find something that is pertinent to commands which complete partial input (e.g., when you type ⟨TAB⟩). If you want to catch index entries that refer to "complete", "completion", and "completing", you could type `icomplet`⟨RET⟩.

Info documents which describe programs should index the commands, options, and key sequences that the program provides. If you are looking for a description of a command, an option, or a key, just type their names when `i` prompts you for a topic. For example, if

you want to read the description of what the `C-f` key does, type `i C - f` ⟨RET⟩. Here `C-f` are 3 literal characters '`C`', '`-`', and '`f`', not the "Control-f" command key you type inside Emacs to run the command bound to `C-f`.

In Emacs, `i` runs the command `Info-index`.

If you don't know what manual documents something, try the `M-x info-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.

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 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).

In Emacs, `Meta-s` is equivalent to `s`. That is for compatibility with other GNU packages that use `M-s` for a similar kind of search command. Both `s` and `M-s` run in Emacs the command `Info-search`.

2.3 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 ⟨undefined⟩ [Menus], page ⟨undefined⟩.

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 or HTML from them. You would use the '`@dircategory`' and '`@direntry`' commands to put the manual into the Info directory. However, if you want to edit an Info file manually and install it manually, 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`' ("formfeed"), or the end of file.[1]

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 '`Expert Info`'. The '`Next`' node is '`Menus`'.

The keywords *Node*, *Next*, *Previous*, and *Up* 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.

[1] 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 '`^_`'.

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 directory for Info files 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)`', it points to a file '`dir`' which holds a large menu listing all the Info documents installed on your site. 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 file name, since when Info searches for a node, it does not expect a file name 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.

2.4 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–what the user must type at the *`m`*'s command prompt 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 in that file directory, but they do not have to; and files in 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.

2.5 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.)
```

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

2.5.1 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 an Info document. So you cannot expect this node to have a '`Next`', '`Previous`' or '`Up`' links 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.

2.6 Quitting Info

To get out of Info, back to what you were doing before, type `q` for *Quit*. This runs `Info-exit` in Emacs.

This is the end of the basic course on using Info. You have learned how to move in an Info document, and how to follow menus and cross references. This makes you ready for reading manuals top to bottom, as new users should do when they learn a new package.

Another set of Info commands is useful when you need to find something quickly in a manual—that is, when you need to use a manual as a reference rather than as a tutorial. We urge you to learn these search commands as well. If you want to do that now, follow this cross reference to ⟨undefined⟩ [Info Search], page ⟨undefined⟩.

Yet another set of commands are meant for experienced users; 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.

2.7 Tags Tables for Info Files

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

To make a tags 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. Info files produced by the `makeinfo` command that is part of the Texinfo package always have tags tables to begin with.

Once the Info file has a tags table, you must make certain it is up to date. If you edit an Info file directly (as opposed to editing its Texinfo source), and, 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 tags table, Info will no longer be able to find that node. To update the tags table, use the `Info-tagify` command again.

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

```
^_^L
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 '`DEL`' character, and the character position in the file of the beginning of the node.

2.8 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.

2.9 Emacs Info-mode Variables

The following variables may modify the behavior 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*. The stand-alone Info reader program has its own set of variables, described in section "Manipulating Variables" in *GNU Info*.

`Info-directory-list`
: The list of directories to search for Info files. Each element is a string (directory name) or `nil` (try default directory). If not initialized Info uses the environment variable `INFOPATH` to initialize it, or `Info-default-directory-list` if there is no `INFOPATH` variable in the environment.

  If you wish to customize the Info directory search list for both Emacs info and stand-alone Info, it is best to set the `INFOPATH` environment variable, since that applies to both programs.

`Info-additional-directory-list`
: A list of additional directories to search for Info documentation files. These directories are not searched for merging the '`dir`' file.

`Info-fontify`
: When set to a non-`nil` value, enables highlighting of Info files. The default is `t`. You can change how the highlighting looks by customizing the faces `info-node`, `info-xref`, `info-header-xref`, `info-header-node`, `info-menu-5`, `info-menu-header`, and `info-title-`*n*`-face` (where *n* is the level of the section, a number between 1 and 4). To customize a face, type *M-x customize-face* ⟨RET⟩ *face* ⟨RET⟩, where *face* is one of the face names listed here.

`Info-use-header-line`
: If non-`nil`, Emacs puts in the Info buffer a header line showing the '`Next`', '`Prev`', and '`Up`' links. A header line does not scroll with the rest of the buffer, making these links always visible.

`Info-hide-note-references`
: As explained in earlier nodes, the Emacs version of Info normally hides some text in menus and cross-references. You can completely disable this feature, by setting this option to `nil`. Setting it to a value that is neither `nil` nor `t` produces an intermediate behavior, hiding a limited amount of text, but showing all text that could potentially be useful.

`Info-scroll-prefer-subnodes`
: If set to a non-`nil` value, ⟨SPC⟩ and ⟨BACKSPACE⟩ (or ⟨DEL⟩) keys in a menu visit subnodes of the current node before scrolling to its end or beginning, respectively. For example, if the node's menu appears on the screen, the next ⟨SPC⟩ moves to a subnode indicated by the following menu item. Setting this option to `nil` results in behavior similar to the stand-alone Info reader program, which visits the first subnode from the menu only when you hit the end of the current node. The default is `nil`.

`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. The Lisp code to be executed should follow the node delimiter (the '`DEL`' character) and an '`execute: `' tag, like this:

```
^_execute: (message "This is an active node!")
```

`Info-enable-edit`
: Set to `nil`, disables the '`e`' (`Info-edit`) command. A non-`nil` value enables it. See ⟨undefined⟩ [Add], page ⟨undefined⟩.

3 Creating an Info File from a Texinfo 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 "Overview of Texinfo" in *Texinfo: The GNU Documentation Format*, to learn how to write a Texinfo file.

See section "Creating an Info File" in *Texinfo: The GNU Documentation Format*, to learn how to create an Info file from a Texinfo file.

See section "Installing an Info File" in *Texinfo: The GNU Documentation Format*, to learn how to install an Info file after you have created one.

Index

This is an alphabetical listing of all the commands, variables, and topics discussed in this document.

(Index is nonexistent)