

gnu:guide/texi

COLLABORATORS

	TITLE : gnu:guide/texi		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		December 11, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	gnu:guide/texi	1
1.1	gnu:guide/texi.guide	1
1.2	texi.guide/Copying	8
1.3	texi.guide/Overview	9
1.4	texi.guide/Using Texinfo	10
1.5	texi.guide/Info Files	11
1.6	texi.guide/Printed Books	12
1.7	texi.guide/Formatting Commands	13
1.8	texi.guide/Conventions	15
1.9	texi.guide/Comments	16
1.10	texi.guide/Minimum	16
1.11	texi.guide/Six Parts	17
1.12	texi.guide/Short Sample	18
1.13	texi.guide/Acknowledgements	20
1.14	texi.guide/Texinfo Mode	21
1.15	texi.guide/Texinfo Mode Overview	21
1.16	texi.guide/Emacs Editing	22
1.17	texi.guide/Inserting	22
1.18	texi.guide/Showing the Structure	25
1.19	texi.guide/Updating Nodes and Menus	25
1.20	texi.guide/Updating Commands	26
1.21	texi.guide/Updating Requirements	28
1.22	texi.guide/Other Updating Commands	29
1.23	texi.guide/Info Formatting	31
1.24	texi.guide/Printing	31
1.25	texi.guide/Texinfo Mode Summary	32
1.26	texi.guide/Beginning a File	35
1.27	texi.guide/Four Parts	35
1.28	texi.guide/Sample Beginning	36
1.29	texi.guide/Header	37

1.30	texi.guide/First Line	38
1.31	texi.guide/Start of Header	38
1.32	texi.guide/setfilename	39
1.33	texi.guide/settitle	39
1.34	texi.guide/setchapternewpage	40
1.35	texi.guide/paragraphindent	41
1.36	texi.guide/End of Header	42
1.37	texi.guide/Info Summary and Permissions	42
1.38	texi.guide/Titlepage & Copyright Page	43
1.39	texi.guide/titlepage	44
1.40	texi.guide/titlefont center sp	45
1.41	texi.guide/title subtitle author	45
1.42	texi.guide/Copyright & Permissions	47
1.43	texi.guide/end titlepage	47
1.44	texi.guide/headings on off	48
1.45	texi.guide/The Top Node	49
1.46	texi.guide/Title of Top Node	50
1.47	texi.guide/Master Menu Parts	50
1.48	texi.guide/Software Copying Permissions	51
1.49	texi.guide/Ending a File	52
1.50	texi.guide/Printing Indices & Menus	52
1.51	texi.guide/Contents	53
1.52	texi.guide/File End	54
1.53	texi.guide/Structuring	55
1.54	texi.guide/Tree Structuring	55
1.55	texi.guide/Structuring Command Types	56
1.56	texi.guide/makeinfo top	57
1.57	texi.guide/chapter	57
1.58	texi.guide/unnumbered & appendix	58
1.59	texi.guide/majorheading & chapheading	58
1.60	texi.guide/section	59
1.61	texi.guide/unnumberedsec appendixsec heading	59
1.62	texi.guide/subsection	60
1.63	texi.guide/unnumberedsubsec appendixsubsec subheading	60
1.64	texi.guide/subsubsection	61
1.65	texi.guide/Nodes	61
1.66	texi.guide/Two Paths	62
1.67	texi.guide/Node Menu Illustration	62
1.68	texi.guide/node	64

1.69	texi.guide/Node Names	65
1.70	texi.guide/Writing a Node	65
1.71	texi.guide/Node Line Tips	66
1.72	texi.guide/Node Line Requirements	67
1.73	texi.guide/First Node	67
1.74	texi.guide/makeinfo top command	68
1.75	texi.guide/Top Node Summary	69
1.76	texi.guide/makeinfo Pointer Creation	69
1.77	texi.guide/Menus	70
1.78	texi.guide/Menu Location	70
1.79	texi.guide/Writing a Menu	71
1.80	texi.guide/Menu Parts	71
1.81	texi.guide/Less Cluttered Menu Entry	72
1.82	texi.guide/Menu Example	72
1.83	texi.guide/Other Info Files	73
1.84	texi.guide/Cross References	74
1.85	texi.guide/References	75
1.86	texi.guide/Cross Reference Commands	75
1.87	texi.guide/Cross Reference Parts	76
1.88	texi.guide/xref	77
1.89	texi.guide/Reference Syntax	78
1.90	texi.guide/One Argument	78
1.91	texi.guide/Two Arguments	79
1.92	texi.guide/Three Arguments	80
1.93	texi.guide/Four and Five Arguments	81
1.94	texi.guide/Top Node Naming	83
1.95	texi.guide/ref	83
1.96	texi.guide/pxref	84
1.97	texi.guide/inforef	85
1.98	texi.guide/Marking Text	86
1.99	texi.guide/Indicating	87
1.100	texi.guide/Useful Highlighting	87
1.101	texi.guide/code	88
1.102	texi.guide/kbd	89
1.103	texi.guide/key	90
1.104	texi.guide/samp	91
1.105	texi.guide/var	92
1.106	texi.guide/file	93
1.107	texi.guide/dfn	93

1.108texi.guide/cite	94
1.109texi.guide/Emphasis	94
1.110texi.guide/emph & strong	95
1.111texi.guide/Smallcaps	95
1.112texi.guide/Fonts	96
1.113texi.guide/Quotations and Examples	96
1.114texi.guide/Block Enclosing Commands	97
1.115texi.guide/quotation	98
1.116texi.guide/example	99
1.117texi.guide/noindent	100
1.118texi.guide/Lisp Example	100
1.119texi.guide/smalexample & smalllisp	101
1.120texi.guide/display	102
1.121texi.guide/format	102
1.122texi.guide/exdent	102
1.123texi.guide/flushleft & flushright	103
1.124texi.guide/cartouche	104
1.125texi.guide/Lists and Tables	104
1.126texi.guide/Introducing Lists	105
1.127texi.guide/itemize	106
1.128texi.guide/enumerate	107
1.129texi.guide/Two-column Tables	109
1.130texi.guide/table	109
1.131texi.guide/ftable vtable	110
1.132texi.guide/itemx	110
1.133texi.guide/Indices	111
1.134texi.guide/Index Entries	112
1.135texi.guide/Predefined Indices	112
1.136texi.guide/Indexing Commands	113
1.137texi.guide/Combining Indices	114
1.138texi.guide/syncodeindex	115
1.139texi.guide/synindex	116
1.140texi.guide/New Indices	116
1.141texi.guide/Insertions	117
1.142texi.guide/Braces Atsigns Periods	117
1.143texi.guide/Inserting An Atsign	118
1.144texi.guide/Inserting Braces	118
1.145texi.guide/Controlling Spacing	118
1.146texi.guide/dmn	119

1.147texi.guide/Dots Bullets	120
1.148texi.guide/dots	120
1.149texi.guide/bullet	120
1.150texi.guide/TeX and copyright	121
1.151texi.guide/tex	121
1.152texi.guide/copyright symbol	121
1.153texi.guide/minus	121
1.154texi.guide/Glyphs	122
1.155texi.guide/Glyphs Summary	122
1.156texi.guide/result	123
1.157texi.guide/expansion	123
1.158texi.guide/Print Glyph	124
1.159texi.guide/Error Glyph	125
1.160texi.guide/Equivalence	125
1.161texi.guide/Point Glyph	126
1.162texi.guide/Breaks	127
1.163texi.guide/Break Commands	127
1.164texi.guide/Line Breaks	128
1.165texi.guide/w	128
1.166texi.guide/sp	129
1.167texi.guide/page	129
1.168texi.guide/group	129
1.169texi.guide/need	130
1.170texi.guide/Definition Commands	131
1.171texi.guide/Def Cmd Template	131
1.172texi.guide/Optional Arguments	133
1.173texi.guide/defnfx	134
1.174texi.guide/Def Cmds in Detail	134
1.175texi.guide/Functions Commands	135
1.176texi.guide/Variables Commands	136
1.177texi.guide/Typed Functions	137
1.178texi.guide/Typed Variables	139
1.179texi.guide/Abstract Objects	140
1.180texi.guide/Data Types	142
1.181texi.guide/Def Cmd Conventions	143
1.182texi.guide/Sample Function Definition	143
1.183texi.guide/Footnotes	145
1.184texi.guide/Conditionals	146
1.185texi.guide/Conditional Commands	147

1.186texi.guide/Using Ordinary TeX Commands	148
1.187texi.guide/set clear value	148
1.188texi.guide/ifset ifclear	149
1.189texi.guide/value	150
1.190texi.guide/value Example	151
1.191texi.guide/Format-Print Hardcopy	152
1.192texi.guide/Use TeX	153
1.193texi.guide/Shell Format & Print	153
1.194texi.guide/Within Emacs	155
1.195texi.guide/Texinfo Mode Printing	156
1.196texi.guide/Compile-Command	157
1.197texi.guide/Requirements Summary	158
1.198texi.guide/Preparing for TeX	158
1.199texi.guide/Overfull hboxes	159
1.200texi.guide/smallbook	160
1.201texi.guide/A4 Paper	161
1.202texi.guide/Cropmarks and Magnification	161
1.203texi.guide/Create an Info File	162
1.204texi.guide/makeinfo advantages	162
1.205texi.guide/Invoking makeinfo	163
1.206texi.guide/makeinfo options	163
1.207texi.guide/Pointer Validation	166
1.208texi.guide/makeinfo in Emacs	166
1.209texi.guide/texinfo-format commands	167
1.210texi.guide/Batch Formatting	168
1.211texi.guide/Tag and Split Files	169
1.212texi.guide/Install an Info File	170
1.213texi.guide/Directory file	170
1.214texi.guide/New Info File	171
1.215texi.guide/Other Info Directories	172
1.216texi.guide/Command List	172
1.217texi.guide/Tips	186
1.218texi.guide/Sample Texinfo File	191
1.219texi.guide/Sample Permissions	193
1.220texi.guide/Inserting Permissions	193
1.221texi.guide/ifinfo Permissions	194
1.222texi.guide/Titlepage Permissions	195
1.223texi.guide/Include Files	195
1.224texi.guide/Using Include Files	196

1.225texi.guide/texinfo-multiple-files-update	197
1.226texi.guide/Include File Requirements	198
1.227texi.guide/Sample Include File	198
1.228texi.guide/Include Files Evolution	199
1.229texi.guide/Headings	200
1.230texi.guide/Headings Introduced	200
1.231texi.guide/Heading Format	201
1.232texi.guide/Heading Choice	202
1.233texi.guide/Custom Headings	203
1.234texi.guide/Catching Mistakes	205
1.235texi.guide/makeinfo preferred	205
1.236texi.guide/Debugging with Info	206
1.237texi.guide/Debugging with TeX	207
1.238texi.guide/Using texinfo-show-structure	209
1.239texi.guide/Using occur	210
1.240texi.guide/Running Info-Validate	211
1.241texi.guide/Using Info-validate	211
1.242texi.guide/Unsplit	212
1.243texi.guide/Tagifying	213
1.244texi.guide/Splitting	213
1.245texi.guide/Refilling Paragraphs	214
1.246texi.guide/Command Syntax	215
1.247texi.guide/Obtaining TeX	216
1.248texi.guide/New Features	216
1.249texi.guide/New Texinfo Mode Commands	217
1.250texi.guide/New Commands	219
1.251texi.guide/Command and Variable Index	223
1.252texi.guide/Concept Index	227

Chapter 1

gnu:guide/texi

1.1 gnu:guide/texi.guide

Texinfo

Texinfo is a documentation system that uses a single source file to produce both on-line information and printed output.

The first part of this master menu lists the major nodes in this Info document, including the @-command and concept indices. The rest of the menu lists all the lower level nodes in the document.

This is Edition 2.18 of the Texinfo documentation, 26 March 1993, for Texinfo Version 2.

Copying	Your rights.
Overview	Texinfo in brief.
Texinfo Mode	How to use Texinfo mode.
Beginning a File	What is at the beginning of a Texinfo file?
Ending a File	What is at the end of a Texinfo file?
Structuring	How to create chapters, sections, subsections, appendices, and other parts.
Nodes	How to write nodes.
Menus	How to write menus.
Cross References	How to write cross references.
Marking Text	How to mark words and phrases as code, keyboard input, meta-syntactic variables, and the like.
Quotations and Examples	How to write quotations, examples, etc.
Lists and Tables	How to write lists and tables.
Indices	How to create indices.
Insertions	How to insert @-signs, braces, etc.
Glyphs	How to indicate results of evaluation, expansion of macros, errors, etc.
Breaks	How to force and prevent line and page breaks.
Definition Commands	How to describe functions and the like in a uniform manner.
Footnotes	How to write footnotes.
Conditionals	How to specify text for either TeX or Info.

Format-Print Hardcopy	How to convert a Texinfo file to a file for printing and how to print that file.
Create an Info File	Convert a Texinfo file into an Info file.
Install an Info File	Make an Info file accessible to users.
Command List	All the Texinfo @-commands.
Tips	Hints on how to write a Texinfo document.
Sample Texinfo File	A sample Texinfo file to look at.
Sample Permissions	Tell readers they have the right to copy and distribute.
Include Files	How to incorporate other Texinfo files.
Headings	How to write page headings and footings.
Catching Mistakes	How to find formatting mistakes.
Refilling Paragraphs	All about paragraph refilling.
Command Syntax	A description of @-Command syntax.
Obtaining TeX	How to Obtain TeX.
New Features	Texinfo second edition features.
Command and Variable Index	A menu containing commands and variables.
Concept Index	A menu covering many topics.

-- The Detailed Node Listing --

Overview of Texinfo

Using Texinfo	Create a conventional printed book or an Info file.
Info Files	What is an Info file?
Printed Books	Characteristics of a printed book or manual.
Formatting Commands	@-commands are used for formatting.
Conventions	General rules for writing a Texinfo file.
Comments	How to write comments and mark regions that the formatting commands will ignore.
Minimum	What a Texinfo file must have.
Six Parts	Usually, a Texinfo file has six parts.
Short Sample	A short sample Texinfo file.
Acknowledgements	

Using Texinfo Mode

Texinfo Mode Overview	How Texinfo mode can help you.
Emacs Editing	Texinfo mode adds to GNU Emacs' general purpose editing features.
Inserting	How to insert frequently used @-commands.
Showing the Structure	How to show the structure of a file.
Updating Nodes and Menus	How to update or create new nodes and menus.
Info Formatting	How to format for Info.
Printing	How to format and print part or all of a file.
Texinfo Mode Summary	Summary of all the Texinfo mode commands.

Updating Nodes and Menus

Updating Commands	Five major updating commands.
Updating Requirements	How to structure a Texinfo file for using the updating command.
Other Updating Commands	How to indent descriptions, insert missing nodes lines, and update nodes in sequence.

Beginning a Texinfo File

Four Parts	Four parts begin a Texinfo file.
Sample Beginning	Here is a sample beginning for a Texinfo file.
Header	The very beginning of a Texinfo file.
Info Summary and Permissions	Summary and copying permissions for Info.
Titlepage & Copyright Page	Creating the title and copyright pages.
The Top Node	Creating the 'Top' node and master menu.
Software Copying Permissions	Ensure that you and others continue to have the right to use and share software.

The Texinfo File Header

First Line	The first line of a Texinfo file.
Start of Header	Formatting a region requires this.
setfilename	Tell Info the name of the Info file.
settitle	Create a title for the printed work.
setchapternewpage	Start chapters on right-hand pages.
paragraphindent	An option to specify paragraph indentation.
End of Header	Formatting a region requires this.

The Title and Copyright Pages

titlepage	Create a title for the printed document.
titlefont center sp	The '@titlefont', '@center', and '@sp' commands.
title subtitle author	The '@title', '@subtitle', and '@author' commands.
Copyright & Permissions	How to write the copyright notice and include copying permissions.
end titlepage	Turn on page headings after the title and copyright pages.
headings on off	An option for turning headings on and off and double or single sided printing.

The 'Top' Node and Master Menu

Title of Top Node	Sketch what the file is about.
Master Menu Parts	A master menu has three or more parts.

Ending a Texinfo File

Printing Indices & Menus	How to print an index in hardcopy and generate index menus in Info.
Contents	How to create a table of contents.
File End	How to mark the end of a file.

Chapter Structuring

Tree Structuring	A manual is like an upside down tree ...
Structuring Command Types	How to divide a manual into parts.
makeinfo top	The '@top' command, part of the 'Top' node.
chapter	
unnumbered & appendix	
majorheading & chapheading	
section	
unnumberedsec appendixsec heading	

subsection
 unnumberedsubsec appendixsubsec subheading
 subsubsection Commands for the lowest level sections.

Nodes

Two Paths Different commands to structure
 Info output and printed output.
 Node Menu Illustration A diagram, and sample nodes and menus.
 node How to write a node, in detail.
 makeinfo Pointer Creation How to create node pointers with 'makeinfo'.

The '@node' Command

Node Names How to choose node and pointer names.
 Writing a Node How to write an '@node' line.
 Node Line Tips Keep names short.
 Node Line Requirements Keep names unique, without @-commands.
 First Node How to write a 'Top' node.
 makeinfo top command How to use the '@top' command.
 Top Node Summary Write a brief description for readers.

Menus

Menu Location Put a menu in a short node.
 Writing a Menu What is a menu?
 Menu Parts A menu entry has three parts.
 Less Cluttered Menu Entry Two part menu entry.
 Menu Example Two and three part menu entries.
 Other Info Files How to refer to a different Info file.

Cross References

References What cross references are for.
 Cross Reference Commands A summary of the different commands.
 Cross Reference Parts A cross reference has several parts.
 xref Begin a reference with 'See' ...
 Top Node Naming How to refer to the beginning of another file.
 ref A reference for the last part of a sentence.
 pxref How to write a parenthetical cross reference.
 info-ref How to refer to an Info-only file.

'@xref'

Reference Syntax What a reference looks like and requires.
 One Argument '@xref' with one argument.
 Two Arguments '@xref' with two arguments.
 Three Arguments '@xref' with three arguments.
 Four and Five Arguments '@xref' with four and five arguments.

Marking Words and Phrases

Indicating How to indicate definitions, files, etc.
 Emphasis How to emphasize text.

Indicating Definitions, Commands, etc.

Useful Highlighting
code
kbd
key
samp
var
file
dfn
cite

Highlighting provides useful information.
How to indicate code.
How to show keyboard input.
How to specify keys.
How to show a literal sequence of characters.
How to indicate a metasyntactic variable.
How to indicate the name of a file.
How to specify a definition.
How to refer to a book that is not in Info.

Emphasizing Text

emph & strong
Smallcaps
Fonts

How to emphasize text in Texinfo.
How to use the small caps font.
Various font commands for printed output.

Quotations and Examples

Block Enclosing Commands

quotation
example
noindent
Lisp Example
smallexample & smalllisp
display
format

exdent
flushleft & flushright
cartouche

Use different constructs for different purposes.
How to write a quotation.
How to write an example in a fixed-width font.
How to prevent paragraph indentation.
How to illustrate Lisp code.
Forms for the '@smallbook' option.
How to write an example in the current font.
How to write an example that does not narrow the margins.
How to undo the indentation of a line.
How to push text flushleft or flushright.
How to draw cartouches around examples.

Making Lists and Tables

Introducing Lists
itemize
enumerate
Two-column Tables

Texinfo formats lists for you.
How to construct a simple list.
How to construct a numbered list.
How to construct a two-column table.

Making a Two-column Table

table
ftable vtable

itemx

How to construct a two-column table.
How to construct a two-column table with automatic indexing.
How to put more entries in the first column.

Creating Indices

Index Entries
Predefined Indices

Indexing Commands
Combining Indices
New Indices

Choose different words for index entries.
Use different indices for different kinds of entry.
How to make an index entry.
How to combine indices.
How to define your own indices.

Combining Indices

syncodeindex	How to merge two indices, using <code>'@code'</code> font for the merged-from index.
synindex	How to merge two indices, using the default font of the merged-to index.

Special Insertions

Braces Atsigns Periods	How to insert braces, <code>'@'</code> and periods.
dmn	How to format a dimension.
Dots Bullets	How to insert dots and bullets.
TeX and copyright	How to insert the TeX logo and the copyright symbol.
minus	How to insert a minus sign.

Inserting `'@'`, Braces, and Periods

Inserting An Atsign	
Inserting Braces	How to insert <code>'{'</code> and <code>'}'</code>
Controlling Spacing	How to insert the right amount of space after punctuation within a sentence.

Inserting Ellipsis, Dots, and Bullets

dots	How to insert dots ...
bullet	How to insert a bullet.

Inserting TeX and the Copyright Symbol

tex	How to insert the TeX logo.
copyright symbol	How to use <code>'@copyright'{}.</code>

Glyphs for Examples

Glyphs Summary	
result	How to show the result of expression.
expansion	How to indicate an expansion.
Print Glyph	How to indicate printed output.
Error Glyph	How to indicate an error message.
Equivalence	How to indicate equivalence.
Point Glyph	How to indicate the location of point.

Making and Preventing Breaks

Break Commands	Cause and prevent splits.
Line Breaks	How to force a single line to use two lines.
w	How to prevent unwanted line breaks.
sp	How to insert blank lines.
page	How to force the start of a new page.
group	How to prevent unwanted page breaks.
need	Another way to prevent unwanted page breaks.

Definition Commands

Def Cmd Template	How to structure a description using a definition command.
Optional Arguments	How to handle optional and repeated arguments.
deffnx	How to group two or more <code>'first'</code> lines.

Def Cmds in Detail	All the definition commands.
Def Cmd Conventions	Conventions for writing definitions.
Sample Function Definition	

The Definition Commands

Functions Commands	Commands for functions and similar entities.
Variables Commands	Commands for variables and similar entities.
Typed Functions	Commands for functions in typed languages.
Typed Variables	Commands for variables in typed languages.
Abstract Objects	Commands for object-oriented programming.
Data Types	The definition command for data types.

Conditionally Visible Text

Conditional Commands	How to specify text for Info or TeX.
Using Ordinary TeX Commands	You can use any and all TeX commands.
set clear value	How to designate which text to format (for both Info and TeX); and how to set a flag to a string that you can insert.

'@set', '@clear', and '@value'

ifset ifclear	Format a region if a flag is set.
value	Replace a flag with a string.
value Example	An easy way to update edition information.

Format and Print Hardcopy

Use TeX	Use TeX to format for hardcopy.
Shell Format & Print	How to format and print a hardcopy manual with shell commands.
Within Emacs	How to format and print from an Emacs shell.
Texinfo Mode Printing	How to format and print in Texinfo mode.
Compile-Command	How to print using Emacs's compile command.
Requirements Summary	TeX formatting requirements summary.
Preparing for TeX	What you need to do to use TeX.
Overfull hboxes	What are and what to do with overfull hboxes.
smallbook	How to print small format books and manuals.
A4 Paper	How to print on European A4 paper.
Cropmarks and Magnification	How to print marks to indicate the size of pages and how to print scaled up output.

Creating an Info File

makeinfo advantages	'makeinfo' provides better error checking.
Invoking makeinfo	How to run 'makeinfo' from a shell.
makeinfo options	Specify fill-column and other options.
Pointer Validation	How to check that pointers point somewhere.
makeinfo in Emacs	How to run 'makeinfo' from Emacs.
texinfo-format commands	Two Info formatting commands written in Emacs Lisp are an alternative to 'makeinfo'.
Batch Formatting	How to format for Info in Emacs Batch mode.
Tag and Split Files	How tagged and split files help Info to run better.

Installing an Info File

Directory file	The top level menu for all Info files.
New Info File	Listing a new info file.
Other Info Directories	How to specify Info files that are located in other directories.

Sample Permissions

Inserting Permissions	How to put permissions in your document.
ifinfo Permissions	Sample 'ifinfo' copying permissions.
Titlepage Permissions	Sample Titlepage copying permissions.

Include Files

Using Include Files	How to use the '@include' command.
texinfo-multiple-files-update	How to create and update nodes and menus when using included files.
Include File Requirements	What 'texinfo-multiple-files-update' expects.
Sample Include File	A sample outer file with included files within it; and a sample included file.
Include Files Evolution	How use of the '@include' command has changed over time.

Page Headings

Headings Introduced	Conventions for using page headings.
Heading Format	Standard page heading formats.
Heading Choice	How to specify the type of page heading.
Custom Headings	How to create your own headings and footings.

Formatting Mistakes

makeinfo preferred	'makeinfo' finds errors.
Debugging with Info	How to catch errors with Info formatting.
Debugging with TeX	How to catch errors with TeX formatting.
Using texinfo-show-structure	How to use 'texinfo-show-structure'.
Using occur	How to list all lines containing a pattern.
Running Info-Validate	How to find badly referenced nodes.

Finding Badly Referenced Nodes

Using Info-validate	How to run 'Info-validate'.
Unsplit	How to create an unsplit file.
Tagifying	How to tagify a file.
Splitting	How to split a file manually.

Second Edition Features

New Texinfo Mode Commands	The updating commands are especially useful.
New Commands	Many newly described @-commands.

1.2 texi.guide/Copying

Texinfo Copying Conditions

The programs currently being distributed that relate to Texinfo include portions of GNU Emacs, plus other separate programs (including 'makeinfo', 'info', 'texindex', and 'texinfo.tex'). These programs are "free"; this means that everyone is free to use them and free to redistribute them on a free basis. The Texinfo-related programs are not in the public domain; they are copyrighted and there are restrictions on their distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of these programs that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to Texinfo, that you receive source code or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of the Texinfo related programs, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the programs that relate to Texinfo. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the licenses for the programs currently being distributed that relate to Texinfo are found in the General Public Licenses that accompany them.

1.3 texi.guide/Overview

Overview of Texinfo

"Texinfo"(1) is a documentation system that uses a single source file to produce both on-line information and printed output. This means that instead of writing two different documents, one for the on-line help or other on-line information and the other for a typeset manual or other printed work, you need write only one document. When the work is revised, you need revise only one document. (You can read the on-line information, known as an "Info file", with an Info documentation-reading program.)

Using Texinfo

Create a conventional printed book

	or an Info file.
Info Files	What is an Info file?
Printed Books	Characteristics of a printed book or manual.
Formatting Commands	@-commands are used for formatting.
Conventions	General rules for writing a Texinfo file.
Comments	How to write comments and mark regions that the formatting commands will ignore.
Minimum	What a Texinfo file must have.
Six Parts	Usually, a Texinfo file has six parts.
Short Sample	A short sample Texinfo file.
Acknowledgements	

----- Footnotes -----

(1) Note that the first syllable of "Texinfo" is pronounced like "speck", not "hex". This odd pronunciation is derived from, but is not the same as, the pronunciation of TeX. In the word TeX, the 'X' is actually the Greek letter "chi" rather than the English letter "ex". Pronounce TeX as if the 'X' were the last sound in the name 'Bach'; but pronounce Texinfo as if the 'x' were a 'k'. Spell "Texinfo" with a capital "T" and write the other letters in lower case.

1.4 texi.guide/Using Texinfo

Using Texinfo

=====

Using Texinfo, you can create a printed document with the normal features of a book, including chapters, sections, cross references, and indices. From the same Texinfo source file, you can create a menu-driven, on-line Info file with nodes, menus, cross references, and indices. You can, if you wish, make the chapters and sections of the printed document correspond to the nodes of the on-line information; and you use the same cross references and indices for both the Info file and the printed work. 'The GNU Emacs Manual' is a good example of a Texinfo file, as is this manual.

To make a printed document, you process a Texinfo source file with the TeX typesetting program. This creates a DVI file that you can typeset and print as a book or report. (Note that the Texinfo language is completely different from TeX's usual language, PlainTeX, which Texinfo replaces.) If you do not have TeX, but do have 'troff' or 'nroff', you can use the 'texi2roff' program instead.

To make an Info file, you process a Texinfo source file with the 'makeinfo' utility or Emacs's 'texinfo-format-buffer' command; this creates an Info file that you can install on-line.

TeX and 'texi2roff' work with many types of printer; similarly, Info works with almost every type of computer terminal. This power makes Texinfo a general purpose system, but brings with it a constraint, which is that a Texinfo file may contain only the customary "typewriter" characters (letters, numbers, spaces, and punctuation marks) but no special graphics.

A Texinfo file is a plain ASCII file containing text and "@-commands" (words preceded by an '@') that tell the typesetting and formatting programs what to do. You may edit a Texinfo file with any text editor; but it is especially convenient to use GNU Emacs since that editor has a special mode, called Texinfo mode, that provides various Texinfo-related features. (See Texinfo Mode.)

Before writing a Texinfo source file, you should become familiar with the Info documentation reading program and learn about nodes, menus, cross references, and the rest. (See info, for more information.)

You can use Texinfo to create both on-line help and printed manuals; moreover, Texinfo is freely redistributable. For these reasons, Texinfo is the format in which documentation for GNU utilities and libraries is written.

1.5 texi.guide/Info Files

Info files
=====

An Info file is a Texinfo file formatted so that the Info documentation reading program can operate on it. ('makeinfo' and 'texinfo-format-buffer' are two commands that convert a Texinfo file into an Info file.)

Info files are divided into pieces called "nodes", each of which contains the discussion of one topic. Each node has a name, and contains both text for the user to read and pointers to other nodes, which are identified by their names. The Info program displays one node at a time, and provides commands with which the user can move to other related nodes.

See info, for more information about using Info.

Each node of an Info file may have any number of child nodes that describe subtopics of the node's topic. The names of child nodes are listed in a "menu" within the parent node; this allows you to use certain Info commands to move to one of the child nodes. Generally, an Info file is organized like a book. If a node is at the logical level of a chapter, its child nodes are at the level of sections; likewise, the child nodes of sections are at the level of subsections.

All the children of any one parent are linked together in a bidirectional chain of 'Next' and 'Previous' pointers. The 'Next' pointer provides a link to the next section, and the 'Previous' pointer provides a link to the previous section. This means that all the nodes that are at the level of sections within a chapter are linked together. Normally the order in this chain is the same as the order of the children in the parent's menu. Each child node records the parent node name as its 'Up' pointer. The last child has no 'Next' pointer, and the first child has the parent both as its 'Previous' and as its 'Up' pointer. (1)

The book-like structuring of an Info file into nodes that correspond to chapters, sections, and the like is a matter of convention, not a requirement. The 'Up', 'Previous', and 'Next' pointers of a node can point to any other nodes, and a menu can contain any other nodes. Thus, the node structure can be any directed graph. But it is usually more comprehensible to follow a structure that corresponds to the structure of chapters and sections in a printed book or report.

In addition to menus and to 'Next', 'Previous', and 'Up' pointers, Info provides pointers of another kind, called references, that can be sprinkled throughout the text. This is usually the best way to represent links that do not fit a hierarchical structure.

Usually, you will design a document so that its nodes match the structure of chapters and sections in the printed output. But there are times when this is not right for the material being discussed. Therefore, Texinfo uses separate commands to specify the node structure for the Info file and the section structure for the printed output.

Generally, you enter an Info file through a node that by convention is called 'Top'. This node normally contains just a brief summary of the file's purpose, and a large menu through which the rest of the file is reached. From this node, you can either traverse the file systematically by going from node to node, or you can go to a specific node listed in the main menu, or you can search the index menus and then go directly to the node that has the information you want.

If you want to read through an Info file in sequence, as if it were a printed manual, you can get the whole file with the advanced Info command 'g* RET'. (See Advanced Info commands.)

The 'dir' file in the 'info' directory serves as the departure point for the whole Info system. From it, you can reach the 'Top' nodes of each of the documents in a complete Info system.

----- Footnotes -----

(1) In some documents, the first child has no 'Previous' pointer. Occasionally, the last child has the node name of the next following higher level node as its 'Next' pointer.

1.6 texi.guide/Printed Books

Printed Books

=====

A Texinfo file can be formatted and typeset as a printed book or manual. To do this, you need TeX, a powerful, sophisticated typesetting program written by Donald Knuth.(1)

A Texinfo-based book is similar to any other typeset, printed work: it can have a title page, copyright page, table of contents, and preface, as well as chapters, numbered or unnumbered sections and

subsections, page headers, cross references, footnotes, and indices.

You can use Texinfo to write a book without ever having the intention of converting it into on-line information. You can use Texinfo for writing a printed novel, and even to write a printed memo, although this latter application is not recommended since electronic mail is so much easier.

TeX is a general purpose typesetting program. Texinfo provides a file called `'texinfo.tex'` that contains information (definitions or "macros") that TeX uses when it typesets a Texinfo file. (`'texinfo.tex'` tells TeX how to convert the Texinfo @-commands to TeX commands, which TeX can then process to create the typeset document.) `'texinfo.tex'` contains the specifications for printing a document.

Most often, documents are printed on 8.5 inch by 11 inch pages (216mm by 280mm; this is the default size), but you can also print for 7 inch by 9.25 inch pages (178mm by 235mm; the `'@smallbook'` size) or on European A4 size paper (`'@afourpaper'`). (See Printing . Also, see See Printing on A4 Paper.)

By changing the parameters in `'texinfo.tex'`, you can change the size of the printed document. In addition, you can change the style in which the printed document is formatted; for example, you can change the sizes and fonts used, the amount of indentation for each paragraph, the degree to which words are hyphenated, and the like. By changing the specifications, you can make a book look dignified, old and serious, or light-hearted, young and cheery.

TeX is freely distributable. It is written in a dialect of Pascal called WEB and can be compiled either in Pascal or (by using a conversion program that comes with the TeX distribution) in C. (See TeX Mode, for information about TeX.)

TeX is very powerful and has a great many features. Because a Texinfo file must be able to present information both on a character-only terminal in Info form and in a typeset book, the formatting commands that Texinfo supports are necessarily limited.

See How to Obtain TeX.

----- Footnotes -----

(1) You can also use the `'texi2roff'` program if you do not have TeX; since Texinfo is designed for use with TeX, `'texi2roff'` is not described here. `'texi2roff'` is part of the standard GNU distribution.

1.7 texi.guide/Formatting Commands

=====

In a Texinfo file, the commands that tell TeX how to typeset the printed manual and tell `'makeinfo'` and `'texinfo-format-buffer'` how to create an Info file are preceded by `'@'`; they are called "`@-commands`".

For example, `@node` is the command to indicate a node and `@chapter` is the command to indicate the start of a chapter.

Please note: All the @-commands, with the exception of the `@TeX{}` command, must be written entirely in lower case.

The Texinfo @-commands are a strictly limited set of constructs. The strict limits make it possible for Texinfo files to be understood both by TeX and by the code that converts them into Info files. You can display Info files on any terminal that displays alphabetic and numeric characters. Similarly, you can print the output generated by TeX on a wide variety of printers.

Depending on what they do or what arguments⁽¹⁾ they take, you need to write @-commands on lines of their own or as part of sentences:

- * Write a command such as `@noindent` at the beginning of a line as the only text on the line. (`@noindent` prevents the beginning of the next line from being indented as the beginning of a paragraph.)
- * Write a command such as `@chapter` at the beginning of a line followed by the command's arguments, in this case the chapter title, on the rest of the line. (`@chapter` creates chapter titles.)
- * Write a command such as `@dots{}` wherever you wish but usually within a sentence. (`@dots{}` creates dots ...)
- * Write a command such as `@code{SAMPLE-CODE}` wherever you wish (but usually within a sentence) with its argument, SAMPLE-CODE in this example, between the braces. (`@code` marks text as being code.)
- * Write a command such as `@example` at the beginning of a line of its own; write the body-text on following lines; and write the matching `@end` command, `@end example` in this case, at the beginning of a line of its own after the body-text. (`@example` ... `@end example` indents and typesets body-text as an example.)

As a general rule, a command requires braces if it mingles among other text; but it does not need braces if it starts a line of its own. The non-alphabetic commands, such as `@:`, are exceptions to the rule; they do not need braces.

As you gain experience with Texinfo, you will rapidly learn how to write the different commands: the different ways to write commands make it easier to write and read Texinfo files than if all commands followed exactly the same syntax. (For details about @-command syntax, see See @-Command Syntax.)

----- Footnotes -----

(1) The word "argument" comes from the way it is used in mathematics and does not refer to a disputation between two people; it refers to the information presented to the command. According to the 'Oxford English Dictionary', the word derives from the Latin for "to make clear, prove"; thus it came to mean 'the evidence offered as

proof', which is to say, 'the information offered', which led to its mathematical meaning. In its other thread of derivation, the word came to mean 'to assert in a manner against which others may make counter assertions', which led to the meaning of 'argument' as a disputation.

1.8 texi.guide/Conventions

General Syntactic Conventions

=====

All ASCII printing characters except '@', '{' and '}' can appear in a Texinfo file and stand for themselves. '@' is the escape character which introduces commands. '{' and '}' should be used only to surround arguments to certain commands. To put one of these special characters into the document, put an '@' character in front of it, like this: '@@', ' '.

It is customary in TeX to use doubled single-quote characters to begin and end quotations: ' ' and ' ' (but without a space between the two single-quote characters). This convention should be followed in Texinfo files. TeX converts doubled single-quote characters to left- and right-hand doubled quotation marks and Info converts doubled single-quote characters to ASCII double-quotes: ' ' and ' ' to " .

Use three hyphens in a row, '---', for a dash--like this. In TeX, a single or even a double hyphen produces a printed dash that is shorter than the usual typeset dash. Info reduces three hyphens to two for display on the screen.

To prevent a paragraph from being indented in the printed manual, put the command '@noindent' on a line by itself before the paragraph.

If you mark off a region of the Texinfo file with the '@iftex' and '@end iftex' commands, that region will appear only in the printed copy; in that region, you can use certain commands borrowed from PlainTeX that you cannot use in Info. Likewise, if you mark off a region with the '@ifinfo' and '@end ifinfo' commands, that region will appear only in the Info file; in that region, you can use Info commands that you cannot use in TeX. (See Conditionals.)

Caution: Do not use tabs in a Texinfo file! TeX uses variable-width fonts, which means that it cannot predefine a tab to work in all circumstances. Consequently, TeX treats tabs like single spaces, and that is not what they look like.

To avoid this problem, Texinfo mode causes GNU Emacs to insert multiple spaces when you press the TAB key.

Also, you can run 'untabify' in Emacs to convert tabs in a region to multiple spaces.

1.9 texi.guide/Comments

Comments

=====

You can write comments in a Texinfo file that will not appear in either the Info file or the printed manual by using the '@comment' command (which may be abbreviated to '@c'). Such comments are for the person who reads the Texinfo file. All the text on a line that follows either '@comment' or '@c' is a comment; the rest of the line does not appear in either the Info file or the printed manual. (Often, you can write the '@comment' or '@c' in the middle of a line, and only the text that follows after the '@comment' or '@c' command does not appear; but some commands, such as '@settitle' and '@setfilename', work on a whole line. You cannot use '@comment' or '@c' in a line beginning with such a command.)

You can write long stretches of text that will not appear in either the Info file or the printed manual by using the '@ignore' and '@end ignore' commands. Write each of these commands on a line of its own, starting each command at the beginning of the line. Text between these two commands does not appear in the processed output. You can use '@ignore' and '@end ignore' for writing comments. Often, '@ignore' and '@end ignore' is used to enclose a part of the copying permissions that applies to the Texinfo source file of a document, but not to the Info or printed version of the document.

1.10 texi.guide/Minimum

What a Texinfo File Must Have

=====

By convention, the names of Texinfo files end with one of the extensions '.texinfo', '.texi', or '.tex'. The longer extension is preferred since it describes more clearly to a human reader the nature of the file. The shorter extensions are for operating systems that cannot handle long file names.

In order to be made into a printed manual and an Info file, a Texinfo file *must* begin with lines like this:

```
\input texinfo
@setfilename INFO-FILE-NAME
@settitle NAME-OF-MANUAL
```

The contents of the file follow this beginning, and then you *must* end a Texinfo file with a line like this:

```
@bye
```

The '\input texinfo' line tells TeX to use the 'texinfo.tex' file, which tells TeX how to translate the Texinfo @-commands into TeX typesetting commands. (Note the use of the backslash, '\\'; this is

correct for TeX.) The '@setfilename' line provides a name for the Info file and the '@settitle' line specifies a title for the page headers (or footers) of the printed manual.

The '@bye' line at the end of the file on a line of its own tells the formatters that the file is ended and to stop formatting.

Usually, you will not use quite such a spare format, but will include mode setting and start-of-header and end-of-header lines at the beginning of a Texinfo file, like this:

```
\input texinfo    @c -*-texinfo*-
@c %**start of header
@setfilename INFO-FILE-NAME
@settitle NAME-OF-MANUAL
@c %**end of header
```

In the first line, ' -*-texinfo-*-' causes Emacs to switch into Texinfo mode when you edit the file.

The '@c' lines which surround the '@setfilename' and '@settitle' lines are optional, but you need them in order to run TeX or Info on just part of the file. (See Start of Header, for more information.)

Furthermore, you will usually provide a Texinfo file with a title page, indices, and the like. But the minimum, which can be useful for short documents, is just the three lines at the beginning and the one line at the end.

1.11 texi.guide/Six Parts

Six Parts of a Texinfo File

=====

Generally, a Texinfo file contains more than the minimal beginning and end--it usually contains six parts:

1. Header

The "Header" names the file, tells TeX which definitions' file to use, and performs other "housekeeping" tasks.

2. Summary Description and Copyright

The "Summary Description and Copyright" segment describes the document and contains the copyright notice and copying permissions for the Info file. The segment must be enclosed between '@ifinfo' and '@end ifinfo' commands so that the formatters place it only in the Info file.

3. Title and Copyright

The "Title and Copyright" segment contains the title and copyright pages and copying permissions for the printed manual. The segment must be enclosed between '@titlepage' and '@end titlepage' commands. The title and copyright page appear only in the printed manual.

4. 'Top' Node and Master Menu

The "Master Menu" contains a complete menu of all the nodes in the whole Info file. It appears only in the Info file, in the 'Top' node.

5. Body

The "Body" of the document may be structured like a traditional book or encyclopedia or it may be free form.

6. End

The "End" contains commands for printing indices and generating the table of contents, and the '@bye' command on a line of its own.

1.12 texi.guide/Short Sample

A Short Sample Texinfo File

=====

Here is a complete but very short Texinfo file, in 6 parts. The first three parts of the file, from '\input texinfo' through to '@end titlepage', look more intimidating than they are. Most of the material is standard boilerplate; when you write a manual, simply insert the names for your own manual in this segment. (See Beginning a File.)

In the following, the sample text is **indented**; comments on it are not. The complete file, without any comments, is shown in See Sample Texinfo File.

Part 1: Header

The header does not appear in either the Info file or the printed output. It sets various parameters, including the name of the Info file and the title used in the header.

```
\input texinfo    @c -*-texinfo-*-
@c %**start of header
@setfilename sample.info
@settitle Sample Document
@c %**end of header
```

```
@setchapternewpage odd
```

Part 2: Summary Description and Copyright

The summary description and copyright segment does not appear in the printed document.

```
@ifinfo
This is a short example of a complete Texinfo file.

Copyright @copyright{} 1990 Free Software Foundation, Inc.
```

```
@end ifinfo
```

Part 3: Titlepage and Copyright

The titlepage segment does not appear in the Info file.

```
@titlepage
@sp 10
@comment The title is printed in a large font.
@center @titlefont{Sample Title}

@c The following two commands start the copyright page.
@page
@vskip 0pt plus 1filll
Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end titlepage
```

Part 4: 'Top' Node and Master Menu

The 'Top' node contains the master menu for the Info file. Since a printed manual uses a table of contents rather than a menu, the master menu appears only in the Info file.

```
@node Top, First Chapter, (dir), (dir)
@comment node-name, next, previous, up

@menu
* First Chapter:: The first chapter is the
                  only chapter in this sample.
* Concept Index:: This index has two entries.
@end menu
```

Part 5: The Body of the Document

The body segment contains all the text of the document, but not the indices or table of contents. This example illustrates a node and a chapter containing an enumerated list.

```
@node First Chapter, Concept Index, Top, Top
@comment node-name, next, previous, up
@chapter First Chapter
@cindex Sample index entry
```

```
This is the contents of the first chapter.
@cindex Another sample index entry
```

Here is a numbered list.

```
@enumerate
@item
This is the first item.

@item
This is the second item.
```

```
@end enumerate
```

The `@code{makeinfo}` and `@code{texinfo-format-buffer}` commands transform a Texinfo file such as this into an Info file; and `@TeX{}` typesets it for a printed manual.

Part 6: The End of the Document

The end segment contains commands both for generating an index in a node and unnumbered chapter of its own and for generating the table of contents; and it contains the `'@bye'` command that marks the end of the document.

```
@node      Concept Index,      , First Chapter, Top
@comment node-name,      next, previous,      up
@unnumbered Concept Index

@printindex cp

@contents
@bye
```

The Results

Here is what the contents of the first chapter of the sample look like:

This is the contents of the first chapter.

Here is a numbered list.

1. This is the first item.
2. This is the second item.

The `'makeinfo'` and `'texinfo-format-buffer'` commands transform a Texinfo file such as this into an Info file; and TeX typesets it for a printed manual.

1.13 texi.guide/Acknowledgements

Acknowledgements

=====

Richard M. Stallman wrote Edition 1.0 of this manual.
Robert J. Chassell revised and extended it, starting with Edition 1.1.

Our thanks go out to all who helped improve this work, particularly to Francois Pinard and David D. Zuhn, who tirelessly recorded and reported mistakes and obscurities; our special thanks go to Melissa Weissshaus for her frequent and often tedious reviews of nearly

similar editions. Our mistakes are our own.

1.14 texi.guide/TeXinfo Mode

Using TeXinfo Mode

You may edit a TeXinfo file with any text editor you choose. A TeXinfo file is no different from any other ASCII file. However, GNU Emacs comes with a special mode, called TeXinfo mode, that provides Emacs commands and tools to help ease your work.

This chapter describes features of GNU Emacs' TeXinfo mode but not any features of the TeXinfo formatting language. If you are reading this manual straight through from the beginning, you may want to skim through this chapter briefly and come back to it after reading succeeding chapters which describe the TeXinfo formatting language in detail.

TeXinfo Mode Overview	How TeXinfo mode can help you.
Emacs Editing	TeXinfo mode adds to GNU Emacs' general purpose editing features.
Inserting	How to insert frequently used @-commands.
Showing the Structure	How to show the structure of a file.
Updating Nodes and Menus	How to update or create new nodes and menus.
Info Formatting	How to format for Info.
Printing	How to format and print part or all of a file.
TeXinfo Mode Summary	Summary of all the TeXinfo mode commands.

1.15 texi.guide/TeXinfo Mode Overview

TeXinfo Mode Overview

=====

TeXinfo mode provides special features for working with TeXinfo files:

- * Insert frequently used @commands.
- * Automatically create '@node' lines.
- * Show the structure of a TeXinfo source file.
- * Automatically create or update the 'Next', 'Previous', and 'Up' pointers of a node.
- * Automatically create or update menus.
- * Automatically create a master menu.

* Format a part or all of a file for Info.

* Typeset and print part or all of a file.

Perhaps the two most helpful features are those for inserting frequently used @-commands and for creating node pointers and menus.

1.16 texi.guide/Emacs Editing

The Usual GNU Emacs Editing Commands

=====

In most cases, the usual Text mode commands work the same in Texinfo mode as they do in Text mode. Texinfo mode adds new editing commands and tools to GNU Emacs' general purpose editing features. The major difference concerns filling. In Texinfo mode, the paragraph separation variable and syntax table are redefined so that Texinfo commands that should be on lines of their own are not inadvertently included in paragraphs. Thus, the 'M-q' ('fill-paragraph') command will refill a paragraph but not mix an indexing command on a line adjacent to it into the paragraph.

In addition, Texinfo mode sets the 'page-delimiter' variable to the value of 'texinfo-chapter-level-regexp'; by default, this is a regular expression matching the commands for chapters and their equivalents, such as appendices. With this value for the page delimiter, you can jump from chapter title to chapter title with the 'C-x]' ('forward-page') and 'C-x [' ('backward-page') commands and narrow to a chapter with the 'C-x p' ('narrow-to-page') command. (See Pages, for details about the page commands.)

You may name a Texinfo file however you wish, but the convention is to end a Texinfo file name with one of the three extensions '.texinfo', '.texi', or '.tex'. A longer extension is preferred, since it is explicit, but a shorter extension may be necessary for operating systems that limit the length of file names. GNU Emacs automatically enters Texinfo mode when you visit a file with a '.texinfo' or '.texi' extension. Also, Emacs switches to Texinfo mode when you visit a file that has '-*-texinfo-*-' in its first line. If ever you are in another mode and wish to switch to Texinfo mode, type 'M-x texinfo-mode'.

Like all other Emacs features, you can customize or enhance Texinfo mode as you wish. In particular, the keybindings are very easy to change. The keybindings described here are the default or standard ones.

1.17 texi.guide/Inserting

Inserting Frequently Used Commands

=====

Texinfo mode provides commands to insert various frequently used keystrokes.

The insert commands are invoked by typing `'C-c'` twice and then the first letter of the `@`-command:

`'C-c C-c c'`

`'M-x texinfo-insert-@code'`

Insert `'@code{'` and put the cursor between the braces.

`'C-c C-c d'`

`'M-x texinfo-insert-@dfn'`

Insert `'@dfn{'` and put the cursor between the braces.

`'C-c C-c e'`

`'M-x texinfo-insert-@end'`

Insert `'@end'` and attempt to insert the correct following word, such as `'example'` or `'table'`. (This command does not handle nested lists correctly, but inserts the word appropriate to the immediately preceding list.)

`'C-c C-c i'`

`'M-x texinfo-insert-@item'`

Insert `'@item'` and put the cursor at the beginning of the next line.

`'C-c C-c k'`

`'M-x texinfo-insert-@kbd'`

Insert `'@kbd{'` and put the cursor between the braces.

`'C-c C-c n'`

`'M-x texinfo-insert-@node'`

Insert `'@node'` and a comment line listing the sequence for the `'Next'`, `'Previous'`, and `'Up'` nodes. Leave point after the `'@node'`.

`'C-c C-c o'`

`'M-x texinfo-insert-@noindent'`

Insert `'@noindent'` and put the cursor at the beginning of the next line.

`'C-c C-c s'`

`'M-x texinfo-insert-@samp'`

Insert `'@samp{'` and put the cursor between the braces.

`'C-c C-c t'`

`'M-x texinfo-insert-@table'`

Insert `'@table'` followed by a SPC and leave the cursor after the SPC.

`'C-c C-c v'`

`'M-x texinfo-insert-@var'`

Insert `'@var{'` and put the cursor between the braces.

```
'C-c C-c x'
'M-x texinfo-insert-@example'
    Insert '@example' and put the cursor at the beginning of the next
    line.

'C-c C-c {'
'M-x texinfo-insert-braces'
    Insert '{}' and put the cursor between the braces.

'C-c C-c }'
'C-c C-c ]'
'M-x up-list'
    Move from between a pair of braces forward past the closing brace.
    Typing 'C-c C-c ]' is easier than typing 'C-c C-c }', which is,
    however, more mnemonic; hence the two keybindings. (Also, you can
    move out from between braces by typing 'C-f'.)
```

To put a command such as '@code{...}' around an *existing* word, position the cursor in front of the word and type 'C-u 1 C-c C-c c'. This makes it easy to edit existing plain text. The value of the prefix argument tells Emacs how many words following point to include between braces--1 for one word, 2 for two words, and so on. Use a negative argument to enclose the previous word or words. If you do not specify a prefix argument, Emacs inserts the @-command string and positions the cursor between the braces. This feature works only for those @-commands that operate on a word or words within one line, such as '@kbd' and '@var'.

This set of insert commands was created after analyzing the frequency with which different @-commands are used in the 'GNU Emacs Manual' and the 'GDB Manual'. If you wish to add your own insert commands, you can bind a keyboard macro to a key, use abbreviations, or extend the code in 'texinfo.el'.

'C-c C-c C-d' ('texinfo-start-menu-description') is an insert command that works differently from the other insert commands. It inserts a node's section or chapter title in the space for the description in a menu entry line. (A menu entry has three parts, the entry name, the node name, and the description. Only the node name is required, but a description helps explain what the node is about. See The Parts of a Menu.)

To use 'texinfo-start-menu-description', position point in a menu entry line and type 'C-c C-c C-d'. The command looks for and copies the title that goes with the node name, and inserts the title as a description; it positions point at beginning of the inserted text so you can edit it. The function does not insert the title if the menu entry line already contains a description.

This command is only an aid to writing descriptions; it does not do the whole job. You must edit the inserted text since a title tends to use the same words as a node name but a useful description uses different words.

1.18 texi.guide/Showing the Structure

Showing the Section Structure of a File

=====

You can show the section structure of a Texinfo file by using the `'C-c C-s'` command (`'texinfo-show-structure'`). This command shows the section structure of a Texinfo file by listing the lines that begin with the `@`-commands for `'@chapter'`, `'@section'`, and the like. It constructs what amounts to a table of contents. These lines are displayed in another buffer called the `'*Occur*'` buffer. In that buffer, you can position the cursor over one of the lines and use the `'C-c C-c'` command (`'occur-mode-goto-occurrence'`), to jump to the corresponding spot in the Texinfo file.

`'C-c C-s'`

`'M-x texinfo-show-structure'`

Show the `'@chapter'`, `'@section'`, and such lines of a Texinfo file.

`'C-c C-c'`

`'M-x occur-mode-goto-occurrence'`

Go to the line in the Texinfo file corresponding to the line under the cursor in the `'*Occur*'` buffer.

If you call `'texinfo-show-structure'` with a prefix argument by typing `'C-u C-c C-s'`, it will list not only those lines with the `'@node'` lines. (This is how the `'texinfo-show-structure'` command worked without an argument in the first version of Texinfo. It was changed because `'@node'` lines clutter up the `'*Occur*'` buffer and are usually not needed.) You can use `'texinfo-show-structure'` with a prefix argument to check whether the `'Next'`, `'Previous'`, and `'Up'` pointers of an `'@node'` line are correct.

Often, when you are working on a manual, you will be interested only in the structure of the current chapter. In this case, you can mark off the region of the buffer that you are interested in with the `'C-x n'` (`'narrow-to-region'`) command and `'texinfo-show-structure'` will work on only that region. To see the whole buffer again, use `'C-x w'` (`'widen'`). (See *Narrowing*, for more information about the narrowing commands.)

In addition to providing the `'texinfo-show-structure'` command, Texinfo mode sets the value of the page delimiter variable to match the chapter-level `@`-commands. This enables you to use the `'C-x]'` (`'forward-page'`) and `'C-x ['` (`'backward-page'`) commands to move forward and backward by chapter, and to use the `'C-x p'` (`'narrow-to-page'`) command to narrow to a chapter. See *Pages*, for more information about the page commands.

1.19 texi.guide/Updating Nodes and Menus

Updating Nodes and Menus

=====

Texinfo mode provides commands for automatically creating or updating menus and node pointers. The commands are called "update" commands because their most frequent use is for updating a Texinfo file after you have worked on it; but you can use them to insert the 'Next', 'Previous', and 'Up' pointers into an '@node' line that has none and to create menus in a file that has none.

If you do not use the updating commands, you need to write menus and node pointers by hand, which is a tedious task.

Updating Commands	Five major updating commands.
Updating Requirements	How to structure a Texinfo file for using the updating command.
Other Updating Commands	How to indent descriptions, insert missing nodes lines, and update nodes in sequence.

1.20 texi.guide/Updating Commands

The Updating Commands

You can use the updating commands

- * to insert or update the 'Next', 'Previous', and 'Up' pointers of a node,
- * to insert or update the menu for a section, and
- * to create a master menu for a Texinfo source file.

You can also use the commands to update all the nodes and menus in a region or in a whole Texinfo file.

The updating commands work only with conventional Texinfo files, which are structured hierarchically like books. In such files, a structuring command line must follow closely after each '@node' line, except for the 'Top' '@node' line. (A "structuring command line" is a line beginning with '@chapter', '@section', or other similar command.)

You can write the structuring command line on the line that follows immediately after an '@node' line or else on the line that follows after a single '@comment' line or a single '@ifinfo' line. You cannot interpose more than one line between the '@node' line and the structuring command line; and you may interpose only an '@comment' line or an '@ifinfo' line.

Commands which work on a whole buffer require that the 'Top' node be followed by a node with an '@chapter' or equivalent-level command. Note that the menu updating commands will not create a main or master menu for a Texinfo file that has only '@chapter'-level nodes! The menu updating commands only create menus *within* nodes for lower level

nodes. To create a menu of chapters, you must provide a 'Top' node.

The menu updating commands remove menu entries that refer to other Info files since they do not refer to nodes within the current buffer. This is a deficiency. Rather than use menu entries, you can use cross references to refer to other Info files. None of the updating commands affect cross references.

Texinfo mode has five updating commands that are used most often: two are for updating the node pointers or menu of a single node (or a region); two are for updating every node pointer and menu in a file; and one, the 'texinfo-master-menu' command, is for creating a master menu for a complete file, and optionally, for updating every node and menu in the whole Texinfo file.

The 'texinfo-master-menu' command is the primary command:

`'C-c C-u m'`

`'M-x texinfo-master-menu'`

Create or update a master menu that includes all the other menus (incorporating the descriptions from pre-existing menus, if any).

With an argument (prefix argument, 'C-u,' if interactive), first create or update all the nodes and all the regular menus in the buffer before constructing the master menu. (See The Top Node and Master Menu, for more about a master menu.)

For 'texinfo-master-menu' to work, the Texinfo file must have a 'Top' node and at least one subsequent node.

After extensively editing a Texinfo file, you can type the following:

```
C-u M-x texinfo-master-menu
or
C-u C-c C-u m
```

This updates all the nodes and menus completely and all at once.

The other major updating commands do smaller jobs and are designed for the person who updates nodes and menus as he or she writes a Texinfo file.

The commands are:

`'C-c C-u C-n'`

`'M-x texinfo-update-node'`

Insert the 'Next', 'Previous', and 'Up' pointers for the node that point is within (i.e., for the '@node' line preceding point). If the '@node' line has pre-existing 'Next', 'Previous', or 'Up' pointers in it, the old pointers are removed and new ones inserted. With an argument (prefix argument, 'C-u', if interactive), this command updates all '@node' lines in the region (which is the text between point and mark).

`'C-c C-u C-m'`

`'M-x texinfo-make-menu'`

Create or update the menu in the node that point is within. With an argument ('C-u' as prefix argument, if interactive), the command makes or updates menus for the nodes which are either within or a part of the region.

Whenever 'texinfo-make-menu' updates an existing menu, the descriptions from that menu are incorporated into the new menu. This is done by copying descriptions from the existing menu to the entries in the new menu that have the same node names. If the node names are different, the descriptions are not copied to the new menu.

'C-c C-u C-e'

'M-x texinfo-every-node-update'

Insert or update the 'Next', 'Previous', and 'Up' pointers for every node in the buffer.

'C-c C-u C-a'

'M-x texinfo-all-menus-update'

Create or update all the menus in the buffer. With an argument ('C-u' as prefix argument, if interactive), first insert or update all the node pointers before working on the menus.

If a master menu exists, the 'texinfo-all-menus-update' command updates it; but the command does not create a new master menu if none already exists. (Use the 'texinfo-master-menu' command for that.)

When working on a document that does not merit a master menu, you can type the following:

```
C-u C-c C-u C-a
or
C-u M-x texinfo-all-menus-update
```

This updates all the nodes and menus.

The 'texinfo-column-for-description' variable specifies the column to which menu descriptions are indented. By default, the value is 32 although it is often useful to reduce it to as low as 24. You can set the variable with the 'M-x edit-options' command (see Editing Variable Values) or with the 'M-x set-variable' command (see Examining and Setting Variables).

Also, the 'texinfo-indent-menu-description' command may be used to indent existing menu descriptions to a specified column. Finally, if you wish, you can use the 'texinfo-insert-node-lines' command to insert missing '@node' lines into a file. (See Other Updating Commands, for more information.)

1.21 texi.guide/Updating Requirements

Updating Requirements

To use the updating commands, you must organize the Texinfo file hierarchically with chapters, sections, subsections, and the like. When you construct the hierarchy of the manual, do not ‘jump down’ more than one level at a time: you can follow the ‘Top’ node with a chapter, but not with a section; you can follow a chapter with a section, but not with a subsection. However, you may ‘jump up’ any number of levels at one time—for example, from a subsection to a chapter.

Each ‘@node’ line, with the exception of the line for the ‘Top’ node, must be followed by a line with a structuring command such as ‘@chapter’, ‘@section’, or ‘@unnumberedsubsec’.

Each ‘@node’ line/structuring-command line combination must look either like this:

```
@node      Comments,  Minimum, Conventions, Overview
@comment   node-name, next,      previous,      up
@section Comments
```

or like this (without the ‘@comment’ line):

```
@node Comments, Minimum, Conventions, Overview
@section Comments
```

In this example, ‘Comments’ is the name of both the node and the section. The next node is called ‘Minimum’ and the previous node is called ‘Conventions’. The ‘Comments’ section is within the ‘Overview’ node, which is specified by the ‘Up’ pointer. (Instead of an ‘@comment’ line, you can write an ‘@ifinfo’ line.)

If a file has a ‘Top’ node, it must be called ‘top’ or ‘Top’ and be the first node in the file.

The menu updating commands create a menu of sections within a chapter, a menu of subsections within a section, and so on. This means that you must have a ‘Top’ node if you want a menu of chapters.

Incidentally, the ‘makeinfo’ command will create an Info file for a hierarchically organized Texinfo file that lacks ‘Next’, ‘Previous’ and ‘Up’ pointers. Thus, if you can be sure that your Texinfo file will be formatted with ‘makeinfo’, you have no need for the ‘update node’ commands. (See Creating an Info File, for more information about ‘makeinfo’.) However, both ‘makeinfo’ and the ‘texinfo-format-...’ commands require that you insert menus in the file.

1.22 texi.guide/Other Updating Commands

Other Updating Commands

In addition to the five major updating commands, Texinfo mode possesses several less frequently used updating commands:

'M-x texinfo-insert-node-lines'

Insert '@node' lines before the '@chapter', '@section', and other sectioning commands wherever they are missing throughout a region in a Texinfo file.

With an argument ('C-u' as prefix argument, if interactive), the 'texinfo-insert-node-lines' command not only inserts '@node' lines but also inserts the chapter or section titles as the names of the corresponding nodes. In addition, it inserts the titles as node names in pre-existing '@node' lines that lack names. Since node names should be more concise than section or chapter titles, you must manually edit node names so inserted.

For example, the following marks a whole buffer as a region and inserts '@node' lines and titles throughout:

```
C-x h C-u M-x texinfo-insert-node-lines
```

(Note that this command inserts titles as node names in '@node' lines; the 'texinfo-start-menu-description' command (see Inserting Frequently Used Commands) inserts titles as descriptions in menu entries, a different action. However, in both cases, you need to edit the inserted text.)

'M-x texinfo-multiple-files-update'

Update nodes and menus in a document built from several separate files. With 'C-u' as a prefix argument, create and insert a master menu in the outer file. With a numeric prefix argument, such as 'C-u 2', first update all the menus and all the 'Next', 'Previous', and 'Up' pointers of all the included files before creating and inserting a master menu in the outer file. The 'texinfo-multiple-files-update' command is described in the appendix on '@include' files. See texinfo-multiple-files-update.

'M-x texinfo-indent-menu-description'

Indent every description in the menu following point to the specified column. You can use this command to give yourself more space for descriptions. With an argument ('C-u' as prefix argument, if interactive), the 'texinfo-indent-menu-description' command indents every description in every menu in the region. However, this command does not indent the second and subsequent lines of a multi-line description.

'M-x texinfo-sequential-node-update'

Insert the names of the nodes immediately following and preceding the current node as the 'Next' or 'Previous' pointers regardless of those nodes' hierarchical level. This means that the 'Next' node of a subsection may well be the next chapter. Sequentially ordered nodes are useful for novels and other documents that you read through sequentially. (However, in Info, the 'g* RET' command lets you look through the file sequentially, so sequentially ordered nodes are not strictly necessary.) With an argument (prefix argument, if interactive), the 'texinfo-sequential-node-update' command sequentially updates all the nodes in the region.

1.23 texi.guide/Info Formatting

Formatting for Info

=====

Texinfo mode provides several commands for formatting part or all of a Texinfo file for Info. Often, when you are writing a document, you want to format only part of a file--that is, a region.

You can use either the `'texinfo-format-region'` or the `'makeinfo-region'` command to format a region:

```
'C-c C-e C-r'
'M-x texinfo-format-region'
'C-c C-m C-r'
'M-x makeinfo-region'
    Format the current region for Info.
```

You can use either the `'texinfo-format-buffer'` or the `'makeinfo-buffer'` command to format a whole buffer:

```
'C-c C-e C-b'
'M-x texinfo-format-buffer'
'C-c C-m C-b'
'M-x makeinfo-buffer'
    Format the current buffer for Info.
```

For example, after writing a Texinfo file, you can type the following:

```
C-u C-c C-u m
or
C-u M-x texinfo-master-menu
```

This updates all the nodes and menus. Then type the following to create an Info file:

```
C-c C-m C-b
or
M-x makeinfo-buffer
```

For the Info formatting commands to work, the file *must* include a line that has `'@setfilename'` in its header.

Not all systems support the `'makeinfo'`-based formatting commands.

See *Create an Info File*, for details about Info formatting.

1.24 texi.guide/Printing

Formatting and Printing

=====

Typesetting and printing a Texinfo file is a multi-step process in which you first create a file for printing (called a DVI file), and then print the file. Optionally, you may also create indices. To do this, you must run the `'texindex'` command after first running the `'tex'` typesetting command; and then you must run the `'tex'` command again.

Often, when you are writing a document, you want to typeset and print only part of a file to see what it will look like. You can use the `'texinfo-tex-region'` and related commands for this purpose. Use the `'texinfo-tex-buffer'` command to format all of a buffer.

```
'C-c C-t C-r'
```

```
'M-x texinfo-tex-region'
```

Run TeX on the region.

```
'C-c C-t C-b'
```

```
'M-x texinfo-tex-buffer'
```

Run TeX on the buffer.

```
'C-c C-t C-i'
```

```
'M-x texinfo-texindex'
```

Run `'texindex'` to sort the indices of a Texinfo file formatted with `'texinfo-tex-region'` or `'texinfo-tex-buffer'`. You must run the `'tex'` command a second time after sorting the raw index files.

```
'C-c C-t C-p'
```

```
'M-x texinfo-tex-print'
```

Print the file (or the part of the file) previously formatted with `'texinfo-tex-buffer'` or `'texinfo-tex-region'`.

For `'texinfo-tex-region'` or `'texinfo-tex-buffer'` to work, the file `*must*` start with a `'\input texinfo'` line and must include an `'@settitle'` line. The file must end with `'@bye'` on a line by itself. (When you use `'texinfo-tex-region'`, you must surround the `'@settitle'` line with start-of-header and end-of-header lines.)

See `Format-Print Hardcopy`, for a description of the other TeX related commands, such as `'tex-show-print-queue'`.

1.25 texi.guide/Texinfo Mode Summary

Texinfo Mode Summary

```
=====
```

In Texinfo mode, each set of commands has default keybindings that begin with the same keys. All the commands that are custom-created for Texinfo mode begin with `'C-c'`. The keys are somewhat mnemonic.

Insert Commands

```
-----
```

The insert commands are invoked by typing `'C-c'` twice and then the first letter of the `@`-command to be inserted. (It might make more sense mnemonically to use `'C-c C-i'`, for `'custom insert'`, but `'C-c C-c'`

is quick to type.)

C-c C-c c	Insert '@code'.
C-c C-c d	Insert '@dfn'.
C-c C-c e	Insert '@end'.
C-c C-c i	Insert '@item'.
C-c C-c n	Insert '@node'.
C-c C-c s	Insert '@samp'.
C-c C-c v	Insert '@var'.
C-c C-c {	Insert braces.
C-c C-c]	
C-c C-c }	Move out of enclosing braces.
C-c C-c C-d	Insert a node's section title in the space for the description in a menu entry line.

Show Structure

The 'texinfo-show-structure' command is often used within a narrowed region.

C-c C-s	List all the headings.
---------	------------------------

The Master Update Command

The 'texinfo-master-menu' command creates a master menu; and can be used to update every node and menu in a file as well.

C-c C-u m	
M-x texinfo-master-menu	Create or update a master menu.

C-u C-c C-u m	With 'C-u' as a prefix argument, first create or update all nodes and regular menus, and then create a master menu.
---------------	---

Update Pointers

The update pointer commands are invoked by typing 'C-c C-u' and then either typing 'C-n' for 'texinfo-update-node' or typing 'C-e' for 'texinfo-every-node-update'.

C-c C-u C-n	Update a node.
C-c C-u C-e	Update every node in the buffer.

Update Menus

Invoke the update menu commands by typing 'C-c C-u' and then either 'C-m' for 'texinfo-make-menu' or 'C-a' for 'texinfo-all-menus-update'. To update both nodes and menus at the same time, precede 'C-c C-u C-a' with 'C-u'.

C-c C-u C-m	Make or update a menu.
C-c C-u C-a	Make or update all menus in a buffer.
C-u C-c C-u C-a	With 'C-u' as a prefix argument, first create or update all nodes and then create or update all menus.

Format for Info

The Info formatting commands that are written in Emacs Lisp are invoked by typing 'C-c C-e' and then either 'C-r' for a region or 'C-b' for the whole buffer.

The Info formatting commands that are written in C and based on the 'makeinfo' program are invoked by typing 'C-c C-m' and then either 'C-r' for a region or 'C-b' for the whole buffer.

Use the 'texinfo-format...' commands:

C-c C-e C-r	Format the region.
C-c C-e C-b	Format the buffer.

Use 'makeinfo':

C-c C-m C-r	Format the region.
C-c C-m C-b	Format the buffer.
C-c C-m C-l	Recenter the 'makeinfo' output buffer.
C-c C-m C-k	Kill the 'makeinfo' formatting job.

Typeset and Print

The TeX typesetting and printing commands are invoked by typing 'C-c C-t' and then another control command: 'C-r' for 'texinfo-tex-region', 'C-b' for 'texinfo-tex-buffer', and so on.

C-c C-t C-r	Run TeX on the region.
C-c C-t C-b	Run TeX on the buffer.
C-c C-t C-i	Run 'texindex'.
C-c C-t C-p	Print the DVI file.
C-c C-t C-q	Show the print queue.
C-c C-t C-d	Delete a job from the print queue.
C-c C-t C-k	Kill the current TeX formatting job.
C-c C-t C-x	Quit a currently stopped TeX formatting job.
C-c C-t C-l	Recenter the output buffer.

Other Updating Commands

The 'other updating commands' do not have standard keybindings because they are rarely used.

M-x texinfo-insert-node-lines	
	Insert missing '@node' lines in region.

With 'C-u' as a prefix argument,
use section titles as node names.

M-x texinfo-multiple-files-update
Update a multi-file document.
With 'C-u 2' as a prefix argument,
create or update all nodes and menus
in all included files first.

M-x texinfo-indent-menu-description
Indent descriptions.

M-x texinfo-sequential-node-update
Insert node pointers in strict sequence.

1.26 texi.guide/Beginning a File

Beginning a Texinfo File

Certain pieces of information must be provided at the beginning of a Texinfo file, such as the name of the file and the title of the document.

Four Parts	Four parts begin a Texinfo file.
Sample Beginning	Here is a sample beginning for a Texinfo file.
Header	The very beginning of a Texinfo file.
Info Summary and Permissions	Summary and copying permissions for Info.
Titlepage & Copyright Page	Creating the title and copyright pages.
The Top Node	Creating the 'Top' node and master menu.
Software Copying Permissions	Ensure that you and others continue to have the right to use and share software.

1.27 texi.guide/Four Parts

Four Parts Begin a File
=====

Generally, the beginning of a Texinfo file has four parts:

1. The header, delimited by special comment lines, that includes the commands for naming the Texinfo file and telling TeX what definitions' file to use when processing the Texinfo file.
2. A short statement of what the file is about, with a copyright notice and copying permissions. This is enclosed in '@ifinfo' and '@end ifinfo' commands so that the formatters place it only in the Info file.

3. A title page and copyright page, with a copyright notice and copying permissions. This is enclosed between '@titlepage' and '@end titlepage' commands. The title and copyright page appear only in the printed manual.
4. The 'Top' node that contains a menu for the whole Info file. The contents of this node appear only in the Info file.

Also, optionally, you may include the copying conditions for a program and a warranty disclaimer. The copying section will be followed by an introduction or else by the first chapter of the manual.

Since the copyright notice and copying permissions for the Texinfo document (in contrast to the copying permissions for a program) are in parts that appear only in the Info file or only in the printed manual, this information must be given twice.

1.28 texi.guide/Sample Beginning

Sample Texinfo File Beginning

=====

The following sample shows what is needed.

```
\input texinfo    @c -*-texinfo*-
@c %**start of header
@setfilename NAME-OF-INFO-FILE
@settitle NAME-OF-MANUAL
@setchapternewpage odd
@c %**end of header

@ifinfo
This file documents ...

Copyright YEAR COPYRIGHT-OWNER

Permission is granted to ...
@end ifinfo

@c This title page illustrates only one of the
@c two methods of forming a title page.

@titlepage
@title NAME-OF-MANUAL-WHEN-PRINTED
@subtitle SUBTITLE-IF-ANY
@subtitle SECOND-SUBTITLE
@author AUTHOR

@c The following two commands
@c start the copyright page.
@page
@vskip 0pt plus 1filll
Copyright @copyright{} YEAR COPYRIGHT-OWNER
```

```

Published by ...

Permission is granted to ...
@end titlepage

@node Top, Overview, (dir), (dir)

@ifinfo
This document describes ...

This document applies to version ...
of the program named ...
@end ifinfo

@menu
* Copying::          Your rights and freedoms.
* First Chapter::    Getting started ...
* Second Chapter::   ...
...
...
@end menu

@node      First Chapter, Second Chapter, top,      top
@comment node-name,      next,      previous, up
@chapter First Chapter
@cindex Index entry for First Chapter

```

1.29 texi.guide/Header

The Texinfo File Header
=====

Texinfo files start with at least three lines that provide Info and TeX with necessary information. These are the `\input texinfo` line, the `'@settitle'` line, and the `'@setfilename'` line. If you want to run TeX on just a part of the Texinfo File, you must write the `'@settitle'` and `'@setfilename'` lines between start-of-header and end-of-header lines.

Thus, the beginning of a Texinfo file looks like this:

```

\input texinfo    @c -*-texinfo-*-
@setfilename sample.info
@settitle Sample Document

```

or else like this:

```

\input texinfo    @c -*-texinfo-*-
@c %**start of header
@setfilename sample.info
@settitle Sample Document
@c %**end of header

```

First Line	The first line of a Texinfo file.
Start of Header	Formatting a region requires this.
setfilename	Tell Info the name of the Info file.
settitle	Create a title for the printed work.
setchapternewpage	Start chapters on right-hand pages.
paragraphindent	An option to specify paragraph indentation.
End of Header	Formatting a region requires this.

1.30 texi.guide/First Line

The First Line of a Texinfo File

Every Texinfo file that is to be the top-level input to TeX must begin with a line that looks like this:

```
\input texinfo    @c -*-texinfo-*
```

This line serves two functions:

1. When the file is processed by TeX, the `\input texinfo` command tells TeX to load the macros needed for processing a Texinfo file. These are in a file called `'texinfo.tex'`, which is usually located in the `'/usr/lib/tex/macros'` directory. TeX uses the backslash, `'\'`, to mark the beginning of a command, just as Texinfo uses `'@'`. The `'texinfo.tex'` file causes the switch from `'\'` to `'@'`; before the switch occurs, TeX requires `'\'`, which is why it appears at the beginning of the file.
2. When the file is edited in GNU Emacs, the `'-*-texinfo-*'` mode specification tells Emacs to use Texinfo mode.

1.31 texi.guide/Start of Header

Start of Header

Write a start-of-header line on the second line of a Texinfo file. Follow the start-of-header line with `'@setfilename'` and `'@settitle'` lines and, optionally, with other command lines, such as `'@smallbook'` or `'@footnotestyle'`; and then by an end-of-header line (see End of Header).

With these lines, you can format part of a Texinfo file for Info or typeset part for printing.

A start-of-header line looks like this:

```
@c %**start of header
```

The odd string of characters, `%**`, is to ensure that no other comment is accidentally taken for a start-of-header line.

1.32 texi.guide/setfilename

```
'@setfilename'  
-----
```

In order to be made into an Info file, a Texinfo file must contain a line that looks like this:

```
@setfilename INFO-FILE-NAME
```

Write the `'@setfilename'` command at the beginning of a line and follow it on the same line by the Info file name. Do not write anything else on the line; anything on the line after the command is considered part of the file name, including a comment.

The `'@setfilename'` line specifies the name of the Info file to be generated. This name should be different from the name of the Texinfo file. The convention is to write a name with a `'.info'` extension, to produce an Info file name such as `'texinfo.info'`.

Some operating systems cannot handle long file names. You can run into a problem even when the file name you specify is itself short enough. This occurs because the Info formatters split a long Info file into short indirect subfiles, and name them by appending `'-1'`, `'-2'`, ..., `'-10'`, `'-11'`, and so on, to the original file name. (See Tag Files and Split Files.) The subfile name `'texinfo.info-10'`, for example, is too long for some systems; so the Info file name for this document is actually `'texinfo'` rather than `'texinfo.info'`.

The Info formatting commands ignore everything written before the `'@setfilename'` line, which is why the very first line of the file (the `'\input'` line) does not need to be commented out. The `'@setfilename'` line is ignored when you typeset a printed manual.

1.33 texi.guide/settitle

```
'@settitle'  
-----
```

In order to be made into a printed manual, a Texinfo file must contain a line that looks like this:

```
@settitle TITLE
```

Write the `'@settitle'` command at the beginning of a line and follow it on the same line by the title. This tells TeX the title to use in a header or footer. Do not write anything else on the line; anything on

the line after the command is considered part of the title, including a comment.

Conventionally, TeX formats a Texinfo file for double-sided output so as to print the title in the left-hand (even-numbered) page headings and the current chapter titles in the right-hand (odd-numbered) page headings. (TeX learns the title of each chapter from each '@chapter' command.) Page footers are not printed.

Even if you are printing in a single-sided style, TeX looks for an '@settitle' command line, in case you include the manual title in the heading.

The '@settitle' command should precede everything that generates actual output in TeX.

Although the title in the '@settitle' command is usually the same as the title on the title page, it does not affect the title as it appears on the title page. Thus, the two do not need not match exactly; and the title in the '@settitle' command can be a shortened or expanded version of the title as it appears on the title page. (See '@titlepage'.)

TeX prints page headings only for that text that comes after the '@end titlepage' command in the Texinfo file, or that comes after an '@headings' command that turns on headings. (See The '@headings' Command, for more information.)

You may, if you wish, create your own, customized headings and footings. See Page Headings, for a detailed discussion of this process.

1.34 texi.guide/setchapternewpage

@setchapternewpage

In a book or a manual, text is usually printed on both sides of the paper, chapters start on right-hand pages, and right-hand pages have odd numbers. But in short reports, text often is printed only on one side of the paper. Also in short reports, chapters sometimes do not start on new pages, but are printed on the same page as the end of the preceding chapter, after a small amount of vertical whitespace.

You can use the '@setchapternewpage' command with various arguments to specify how TeX should start chapters and whether it should typeset pages for printing on one or both sides of the paper (single-sided or double-sided printing).

Write the '@setchapternewpage' command at the beginning of a line followed by its argument.

For example, you would write the following to cause each chapter to start on a fresh odd-numbered page:

```
@setchapternewpage odd
```

You can specify one of three alternatives with the `'@setchapternewpage'` command:

```
'@setchapternewpage off'
```

Cause TeX to typeset a new chapter on the same page as the last chapter, after skipping some vertical whitespace. Also, cause TeX to format page headers for single-sided printing. (You can override the headers format with the `'@headings double'` command; see See The `'@headings'` Command.)

```
'@setchapternewpage on'
```

Cause TeX to start new chapters on new pages and to typeset page headers for single-sided printing. This is the form most often used for short reports.

This alternative is the default.

```
'@setchapternewpage odd'
```

Cause TeX to start new chapters on new, odd-numbered pages (right-handed pages) and to typeset for double-sided printing. This is the form most often used for books and manuals.

Texinfo does not have an `'@setchapternewpage even'` command.

(You can countermand or modify an `'@setchapternewpage'` command with an `'@headings'` command. See The `'@headings'` Command.)

At the beginning of a manual or book, pages are not numbered--for example, the title and copyright pages of a book are not numbered. By convention, table of contents pages are numbered with roman numerals and not in sequence with the rest of the document.

Since an Info file does not have pages, the `'@setchapternewpage'` command has no effect on it.

Usually, you do not write an `'@setchapternewpage'` command for single-sided printing, but accept the default which is to typeset for single-sided printing and to start new chapters on new pages. Usually, you write an `'@setchapternewpage odd'` command for double-sided printing.

1.35 texi.guide/paragraphindent

Paragraph Indenting

The Info formatting commands may insert spaces at the beginning of the first line of each paragraph, thereby indenting that paragraph. You can use the `'@paragraphindent'` command to specify the indentation. Write an `'@paragraphindent'` command at the beginning of a line followed by either `'asis'` or a number. The template is:

```
@paragraphindent INDENT
```

The Info formatting commands indent according to the value of `INDENT`:

- * If the value of `INDENT` is `'asis'`, the Info formatting commands do not change the existing indentation.
- * If the value of `INDENT` is 0, the Info formatting commands delete existing indentation.
- * If the value of `INDENT` is greater than 0, the Info formatting commands indent the paragraph by that number of spaces.

The default value of `INDENT` is `'asis'`.

Write the `'@paragraphindent'` command before or shortly after the end-of-header line at the beginning of a Texinfo file. (If you write the command between the start-of-header and end-of-header lines, the region formatting commands indent paragraphs as specified.)

A peculiarity of `'texinfo-format-buffer'` and `'texinfo-format-region'` is that they do not indent (nor fill) paragraphs that contain `'@w'` or `'@*'` commands. See Refilling Paragraphs, for a detailed description of what goes on.

1.36 texi.guide/End of Header

End of Header

Follow the header lines with an end-of-header line. An end-of-header line looks like this:

```
@c %**end of header
```

If you include the `'@setchapternewpage'` command between the start-of-header and end-of-header lines, TeX will typeset a region as that command specifies. Similarly, if you include an `'@smallbook'` command between the start-of-header and end-of-header lines, TeX will typeset a region in the "small" book format.

The reason for the odd string of characters (`'%**'`) is so that the `'texinfo-tex-region'` command does not accidentally find something that it should not when it is looking for the header.

The start-of-header line and the end-of-header line are Texinfo mode variables that you can change.

1.37 texi.guide/Info Summary and Permissions

Summary and Copying Permissions for Info

=====

The title page and the copyright page appear only in the printed copy of the manual; therefore, the same information must be inserted in a section that appears only in the Info file. This section usually contains a brief description of the contents of the Info file, a copyright notice, and copying permissions.

The copyright notice should read:

Copyright YEAR COPYRIGHT-OWNER

and be put on a line by itself.

Standard text for the copyright permissions is contained in an appendix to this manual; see See 'ifinfo' Copying Permissions, for the complete text.

The permissions text appears in an Info file **before** the first node. This mean that a reader does **not** see this text when reading the file using Info, except when using the advanced Info command 'g *'.

1.38 texi.guide/Titlepage & Copyright Page

The Title and Copyright Pages

=====

A manual's name and author are usually printed on a title page. Sometimes copyright information is printed on the title page as well; more often, copyright information is printed on the back of the title page.

The title and copyright pages appear in the printed manual, but not in the Info file. Because of this, it is possible to use several slightly obscure TeX typesetting commands that cannot be used in an Info file. In addition, this part of the beginning of a Texinfo file contains the text of the copying permissions that will appear in the printed manual.

See Titlepage Copying Permissions, for the standard text for the copyright permissions.

titlepage	Create a title for the printed document.
titlefont center sp	The '@titlefont', '@center', and '@sp' commands.
title subtitle author	The '@title', '@subtitle', and '@author' commands.
Copyright & Permissions	How to write the copyright notice and include copying permissions.
end titlepage	Turn on page headings after the title and copyright pages.

headings on off

An option for turning headings on and off
and double or single sided printing.

1.39 texi.guide/titlepage

`'@titlepage'`

Start the material for the title page and following copyright page with `'@titlepage'` on a line by itself and end it with `'@end titlepage'` on a line by itself.

The `'@end titlepage'` command starts a new page and turns on page numbering. (See Page Headings, for details about how to generate of page headings.) All the material that you want to appear on unnumbered pages should be put between the `'@titlepage'` and `'@end titlepage'` commands. By using the `'@page'` command you can force a page break within the region delineated by the `'@titlepage'` and `'@end titlepage'` commands and thereby create more than one unnumbered page. This is how the copyright page is produced. (The `'@titlepage'` command might perhaps have been better named the `'@titleandadditionalpages'` command, but that would have been rather long!)

When you write a manual about a computer program, you should write the version of the program to which the manual applies on the title page. If the manual changes more frequently than the program or is independent of it, you should also include an edition number(1) for the manual. This helps readers keep track of which manual is for which version of the program. (The `'Top'` node should also contain this information; see See `'@top'`.)

Texinfo provides two methods for creating a title page. One method uses the `'@titlefont'`, `'@sp'`, and `'@center'` commands to generate a title page in which the words on the page are centered.

The second method uses the `'@title'`, `'@subtitle'`, and `'@author'` commands to create a title page with black rules under the title and author lines and the subtitle text set flush to the right hand side of the page. With this method, you do not specify any of the actual formatting of the title page. You specify the text you want, and Texinfo does the formatting. You may use either method.

----- Footnotes -----

(1) We have found that it is helpful to refer to versions of manuals as `'editions'` and versions of programs as `'versions'`; otherwise, we find we are liable to confuse each other in conversation by referring to both the documentation and the software with the same words.

1.40 texi.guide/titlefont center sp

`'@titlefont', '@center', and '@sp'`

You can use the `'@titlefont'`, `'@sp'`, and `'@center'` commands to create a title page for a printed document. (This is the first of the two methods for creating a title page in Texinfo.)

Use the `'@titlefont'` command to select a large font suitable for the title itself.

For example:

```
@titlefont{Texinfo}
```

Use the `'@center'` command at the beginning of a line to center the remaining text on that line. Thus,

```
@center @titlefont{Texinfo}
```

centers the title, which in this example is "Texinfo" printed in the title font.

Use the `'@sp'` command to insert vertical space. For example:

```
@sp 2
```

This inserts two blank lines on the printed page. (See `'@sp'`, for more information about the `'@sp'` command.)

A template for this method looks like this:

```
@titlepage
@sp 10
@center @titlefont{NAME-OF-MANUAL-WHEN-PRINTED}
@sp 2
@center SUBTITLE-IF-ANY
@sp 2
@center AUTHOR
...
@end titlepage
```

The spacing of the example fits an 8 1/2 by 11 inch manual.

1.41 texi.guide/title subtitle author

`'@title', '@subtitle', and '@author'`

You can use the `'@title'`, `'@subtitle'`, and `'@author'` commands to create a title page in which the vertical and horizontal spacing is done for you automatically. This contrasts with the method described in

the previous section, in which the `'@sp'` command is needed to adjust vertical spacing.

Write the `'@title'`, `'@subtitle'`, or `'@author'` commands at the beginning of a line followed by the title, subtitle, or author.

The `'@title'` command produces a line in which the title is set flush to the left-hand side of the page in a larger than normal font. The title is underlined with a black rule.

The `'@subtitle'` command sets subtitles in a normal-sized font flush to the right-hand side of the page.

The `'@author'` command sets the names of the author or authors in a middle-sized font flush to the left-hand side of the page on a line near the bottom of the title page. The names are underlined with a black rule that is thinner than the rule that underlines the title. (The black rule only occurs if the `'@author'` command line is followed by an `'@page'` command line.)

There are two ways to use the `'@author'` command: you can write the name or names on the remaining part of the line that starts with an `'@author'` command:

```
@author by Jane Smith and John Doe
```

or you can write the names one above each other by using two (or more) `'@author'` commands:

```
@author Jane Smith
@author John Doe
```

(Only the bottom name is underlined with a black rule.)

A template for this method looks like this:

```
@titlepage
@title NAME-OF-MANUAL-WHEN-PRINTED
@subtitle SUBTITLE-IF-ANY
@subtitle SECOND-SUBTITLE
@author AUTHOR
@page
...
@end titlepage
```

Contrast this form with the form of a title page written using the `'@sp'`, `'@center'`, and `'@titlefont'` commands:

```
@titlepage
@sp 10
@center @titlefont{Name of Manual When Printed}
@sp 2
@center Subtitle, If Any
@sp 1
@center Second subtitle
@sp 2
@center Author
```

```
@page
...
@end titlepage
```

1.42 texi.guide/Copyright & Permissions

Copyright Page and Permissions

By international treaty, the copyright notice for a book should be either on the title page or on the back of the title page. The copyright notice should include the year followed by the name of the organization or person who owns the copyright.

When the copyright notice is on the back of the title page, that page is customarily not numbered. Therefore, in Texinfo, the information on the copyright page should be within '@titlepage' and '@end titlepage' commands.

Use the '@page' command to cause a page break. To push the copyright notice and the other text on the copyright page towards the bottom of the page, you can write a somewhat mysterious line after the '@page' command that reads like this:

```
@vskip 0pt plus 1filll
```

This is a TeX command that is not supported by the Info formatting commands. The '@vskip' command inserts whitespace. The '0pt plus 1filll' means to put in zero points of mandatory whitespace, and as much optional whitespace as needed to push the following text to the bottom of the page. Note the use of three 'l's in the word 'filll'; this is the correct usage in TeX.

In a printed manual, the '@copyright{ }' command generates a 'c' inside a circle. (In Info, it generates '(C)'.) The copyright notice itself has the following legally defined sequence:

```
Copyright (C) YEAR COPYRIGHT-OWNER
```

It is customary to put information on how to get a manual after the copyright notice, followed by the copying permissions for the manual.

Note that permissions must be given here as well as in the summary segment within '@ifinfo' and '@end ifinfo' that immediately follows the header since this text appears only in the printed manual and the 'ifinfo' text appears only in the Info file.

See Sample Permissions, for the standard text.

1.43 texi.guide/end titlepage

Heading Generation

An `'@end titlepage'` command on a line by itself not only marks the end of the title and copyright pages, but also causes TeX to start generating page headings and page numbers.

To repeat what is said elsewhere, Texinfo has two standard page heading formats, one for documents which are printed on one side of each sheet of paper (single-sided printing), and the other for documents which are printed on both sides of each sheet (double-sided printing). (See `'@setchapternewpage'`.) You can specify these formats in different ways:

- * The conventional way is to write an `'@setchapternewpage'` command before the title page commands, and then have the `'@end titlepage'` command start generating page headings in the manner desired. (See `'@setchapternewpage'`.)
- * Alternatively, you can use the `'@headings'` command to prevent page headings from being generated or to start them for either single or double-sided printing. (Write an `'@headings'` command immediately after the `'@end titlepage'` command. See The `'@headings'` Command, for more information.)
- * Or, you may specify your own page heading and footing format. See Page Headings, for detailed information about page headings and footings.

Most documents are formatted with the standard single-sided or double-sided format, using `'@setchapternewpage odd'` for double-sided printing and no `'@setchapternewpage'` command for single-sided printing.

1.44 texi.guide/headings on off

The `'@headings'` Command

The `'@headings'` command is rarely used. It specifies what kind of page headings and footings to print on each page. Usually, this is controlled by the `'@setchapternewpage'` command. You need the `'@headings'` command only if the `'@setchapternewpage'` command does not do what you want, or if you want to turn off pre-defined page headings prior to defining your own. Write an `'@headings'` command immediately after the `'@end titlepage'` command.

There are four ways to use the `'@headings'` command:

`'@headings off'`

Turn off printing of page headings.

`'@headings single'`

Turn on page headings appropriate for single-sided printing.

```
\@headings double'
```

```
\@headings on'
```

Turn on page headings appropriate for double-sided printing. The two commands, '@headings on' and '@headings double', are synonymous.

For example, suppose you write '@setchapternewpage off' before the '@titlepage' command to tell TeX to start a new chapter on the same page as the end of the last chapter. This command also causes TeX to typeset page headers for single-sided printing. To cause TeX to typeset for double sided printing, write '@headings double' after the '@end titlepage' command.

You can stop TeX from generating any page headings at all by writing '@headings off' on a line of its own immediately after the line containing the '@end titlepage' command, like this:

```
@end titlepage
@headings off
```

The '@headings off' command overrides the '@end titlepage' command, which would otherwise cause TeX to print page headings.

You can also specify your own style of page heading and footing. See Page Headings, for more information.

1.45 texi.guide/The Top Node

The 'Top' Node and Master Menu

```
=====
```

The 'Top' node is the node from which you enter an Info file.

A 'Top' node should contain a brief description of the Info file and an extensive, master menu for the whole Info file. This helps the reader understand what the Info file is about. Also, you should write the version number of the program to which the Info file applies; or, at least, the edition number.

The contents of the 'Top' node should appear only in the Info file; none of it should appear in printed output, so enclose it between '@ifinfo' and '@end ifinfo' commands. (TeX does not print either an '@node' line or a menu; they appear only in Info; strictly speaking, you are not required to enclose these parts between '@ifinfo' and '@end ifinfo', but it is simplest to do so. See Conditionally Visible Text.)

Title of Top Node
Master Menu Parts

Sketch what the file is about.
A master menu has three or more parts.

1.46 texi.guide/Title of Top Node

'Top' Node Title

Sometimes, you will want to place an '@top' sectioning command line containing the title of the document immediately after the '@node Top' line (see The '@top' Sectioning Command, for more information).

For example, the beginning of the Top node of this manual contains an '@top' sectioning command, a short description, and edition and version information. It looks like this:

```
...
@end titlepage

@ifinfo
@node Top, Copying, (dir), (dir)
@top Texinfo

Texinfo is a documentation system...

This is edition...
...
@end ifinfo

@menu
* Copying::                Texinfo is freely
                           redistributable.
* Overview::              What is Texinfo?
...
@end menu
```

In a 'Top' node, the 'Previous', and 'Up' nodes usually refer to the top level directory of the whole Info system, which is called '(dir)'. The 'Next' node refers to the first node that follows the main or master menu, which is usually the copying permissions, introduction, or first chapter.

1.47 texi.guide/Master Menu Parts

Parts of a Master Menu

A "master menu" is a detailed main menu listing all the nodes in a file.

A master menu is enclosed in '@menu' and '@end menu' commands and does not appear in the printed document.

Generally, a master menu is divided into parts.

* The first part contains the major nodes in the Texinfo file: the

nodes for the chapters, chapter-like sections, and the appendices.

- * The second part contains nodes for the indices.
- * The third and subsequent parts contain a listing of the other, lower level nodes, often ordered by chapter. This way, rather than go through an intermediary menu, an inquirer can go directly to a particular node when searching for specific information. These menu items are not required; add them if you think they are a convenience.

Each section in the menu can be introduced by a descriptive line. So long as the line does not begin with an asterisk, it will not be treated as a menu entry. (See Writing a Menu, for more information.)

For example, the master menu for this manual looks like the following (but has many more entries):

```
@menu
* Copying::          Texinfo is freely
                     redistributable.
* Overview::         What is Texinfo?
* Texinfo Mode::     Special features in GNU Emacs.
...
...
* Command and Variable Index::
                     An entry for each @-command.
* Concept Index::    An entry for each concept.

--- The Detailed Node Listing ---

Overview of Texinfo

* Info Files::       What is an Info file?
* Printed Manuals::  Characteristics of
                     a printed manual.
...
...

Using Texinfo Mode

* Info on a Region::  Formatting part of a file
                     for Info.
...
...
@end menu
```

1.48 texi.guide/Software Copying Permissions

Software Copying Permissions

=====

If the Texinfo file has a section containing the "General Public License" and the distribution information and a warranty disclaimer for

the software that is documented, this section usually follows the 'Top' node. The General Public License is very important to Project GNU software. It ensures that you and others will continue to have a right to use and share the software.

The copying and distribution information and the disclaimer are followed by an introduction or else by the first chapter of the manual.

Although an introduction is not a required part of a Texinfo file, it is very helpful. Ideally, it should state clearly and concisely what the file is about and who would be interested in reading it. In general, an introduction would follow the licensing and distribution information, although sometimes people put it earlier in the document. Usually, an introduction is put in an '@unnumbered' section. (See The '@unnumbered' and '@appendix' Commands.)

1.49 texi.guide/Ending a File

Ending a Texinfo File

The end of a Texinfo file should include the commands that create indices and generate detailed and summary tables of contents. And it must include the '@bye' command that marks the last line processed by TeX.

For example:

```
@node      Concept Index,      , Variables Index, Top
@c         node-name,      next, previous,      up
@unnumbered Concept Index

@printindex cp

@contents
@bye
```

Printing Indices & Menus	How to print an index in hardcopy and generate index menus in Info.
Contents	How to create a table of contents.
File End	How to mark the end of a file.

1.50 texi.guide/Printing Indices & Menus

Index Menus and Printing an Index

=====

To print an index means to include it as part of a manual or Info file. This does not happen automatically just because you use

'@cindex' or other index-entry generating commands in the Texinfo file; those just cause the raw data for the index to be accumulated. To generate an index, you must include the '@printindex' command at the place in the document where you want the index to appear. Also, as part of the process of creating a printed manual, you must run a program called 'texindex' (see Format-Print Hardcopy) to sort the raw data to produce a sorted index file. The sorted index file is what is actually used to print the index.

Texinfo offers six different types of predefined index: the concept index, the function index, the variables index, the keystroke index, the program index, and the data type index (see Predefined Indices). Each index type has a two-letter name: 'cp', 'fn', 'vr', 'ky', 'pg', and 'tp'. You may merge indices, or put them into separate sections (see Combining Indices); or you may define your own indices (see Defining New Indices).

The '@printindex' command takes a two-letter index name, reads the corresponding sorted index file and formats it appropriately into an index.

The '@printindex' command does not generate a chapter heading for the index. Consequently, you should precede the '@printindex' command with a suitable section or chapter command (usually '@unnumbered') to supply the chapter heading and put the index into the table of contents. Precede the '@unnumbered' command with an '@node' line.

For example:

```
@node Variable Index, Concept Index, Function Index, Top
@comment      node-name,          next,          previous, up
@unnumbered Variable Index

@printindex vr

@node      Concept Index,          , Variable Index, Top
@comment      node-name, next,          previous, up
@unnumbered Concept Index

@printindex cp

@summarycontents
@contents
@bye
```

(Readers often prefer that the concept index come last in a book, since that makes it easiest to find.)

1.51 texi.guide/Contents

Generating a Table of Contents

=====

The '@chapter', '@section', and other structuring commands supply

the information to make up a table of contents, but they do not cause an actual table to appear in the manual. To do this, you must use the '@contents' and '@summarycontents' commands:

@contents'

Generate a table of contents in a printed manual, including all chapters, sections, subsections, etc., as well as appendices and unnumbered chapters. (Headings generated by the '@heading' series of commands do not appear in the table of contents.) The '@contents' command should be written on a line by itself.

@shortcontents'

@summarycontents'

('@summarycontents' is a synonym for '@shortcontents'; the two commands are exactly the same.)

Generate a short or summary table of contents that lists only the chapters (and appendices and unnumbered chapters). Omit sections, subsections and subsubsections. Only a long manual needs a short table of contents in addition to the full table of contents.

Write the '@shortcontents' command on a line by itself right *before* the '@contents' command.

The table of contents commands automatically generate a chapter-like heading at the top of the first table of contents page. Write the table of contents commands at the very end of a Texinfo file, just before the '@bye' command, following any index sections--anything in the Texinfo file after the table of contents commands will be omitted from the table of contents.

When you print a manual with a table of contents, the table of contents are printed last and numbered with roman numerals. You need to place those pages in their proper place, after the title page, yourself. (This is the only collating you need to do for a printed manual. The table of contents is printed last because it is generated after the rest of the manual is typeset.)

Here is an example of where to write table of contents commands:

```
INDICES...
@shortcontents
@contents
@bye
```

Since an Info file uses menus instead of tables of contents, the Info formatting commands ignore the '@contents' and '@shortcontents' commands.

1.52 texi.guide/File End

@bye' File Ending

=====

An '@bye' command terminates TeX or Info formatting. None of the formatting commands see any of the file following '@bye'. The '@bye' command should be on a line by itself.

If you wish, you may follow the '@bye' line with notes. These notes will not be formatted and will not appear in either Info or a printed manual; it is as if text after '@bye' were within '@ignore' ... '@end ignore'. Also, you may follow the '@bye' line with a local variables list. See Using Local Variables and the Compile Command, for more information.

1.53 texi.guide/Structuring

Chapter Structuring

The "chapter structuring" commands divide a document into a hierarchy of chapters, sections, subsections, and subsubsections. These commands generate large headings; they also provide information for the table of contents of a printed manual (see Generating a Table of Contents).

The chapter structuring commands do not create an Info node structure, so normally you should put an '@node' command immediately before each chapter structuring command (see Nodes). The only time you are likely to use the chapter structuring commands without using the node structuring commands is if you are writing a document that contains no cross references and will never be transformed into Info format.

It is unlikely that you will ever write a Texinfo file that is intended only as an Info file and not as a printable document. If you do, you might still use chapter structuring commands to create a heading at the top of each node--but you don't need to.

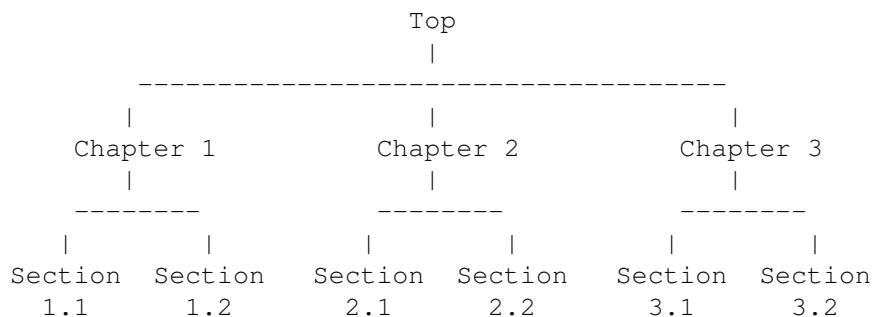
Tree Structuring	A manual is like an upside down tree ...
Structuring Command Types	How to divide a manual into parts.
makeinfo top	The '@top' command, part of the 'Top' node.
chapter	
unnumbered & appendix	
majorheading & chapheading	
section	
unnumberedsec appendixsec heading	
subsection	
unnumberedsubsec appendixsubsec subheading	
subsubsection	Commands for the lowest level sections.

1.54 texi.guide/Tree Structuring

Tree Structure of Sections

A Texinfo file is usually structured like a book with chapters, sections, subsections, and the like. This structure can be visualized as a tree (or rather as an upside-down tree) with the root at the top and the levels corresponding to chapters, sections, subsection, and subsubsections.

Here is a diagram that shows a Texinfo file with three chapters, each of which has two sections.



In a Texinfo file that has this structure, the beginning of Chapter 2 looks like this:

```
@node    Chapter 2, Chapter 3, Chapter 1, top
@chapter Chapter 2
```

The chapter structuring commands are described in the sections that follow; the '@node' and '@menu' commands are described in following chapters. (See Nodes, and see See Menus.)

1.55 texi.guide/Structuring Command Types

Types of Structuring Command

The chapter structuring commands fall into four groups or series, each of which contains structuring commands corresponding to the hierarchical levels of chapters, sections, subsections, and subsubsections.

The four groups are the '@chapter' series, the '@unnumbered' series, the '@appendix' series, and the '@heading' series.

Each command produces titles that have a different appearance on the printed page or Info file; only some of the commands produce titles that are listed in the table of contents of a printed book or manual.

- * The '@chapter' and '@appendix' series of commands produce numbered or lettered entries both in the body of a printed work and in its table of contents.

- * The `'@unnumbered'` series of commands produce unnumbered entries both in the body of a printed work and in its table of contents. The `'@top'` command, which has a special use, is a member of this series (see `'@top'`).
- * The `'@heading'` series of commands produce unnumbered headings that do not appear in a table of contents. The heading commands never start a new page.
- * The `'@majorheading'` command produces results similar to using the `'@chapheading'` command but generates a larger vertical whitespace before the heading.
- * When an `'@setchapternewpage'` command says to do so, the `'@chapter'`, `'@unnumbered'`, and `'@appendix'` commands start new pages in the printed manual; the `'@heading'` commands do not.

Here are the four groups of chapter structuring commands:

Numbered In contents	Unnumbered In contents	Lettered and numbered In contents	No new pages Unnumbered Not in contents
	<code>@top</code>		<code>@majorheading</code>
<code>@chapter</code>	<code>@unnumbered</code>	<code>@appendix</code>	<code>@chapheading</code>
<code>@section</code>	<code>@unnumberedsec</code>	<code>@appendixsec</code>	<code>@heading</code>
<code>@subsection</code>	<code>@unnumberedsubsec</code>	<code>@appendixsubsec</code>	<code>@subheading</code>
<code>@subsubsection</code>	<code>@unnumberedsubsubsec</code>	<code>@appendixsubsubsec</code>	<code>@subsubheading</code>

1.56 texi.guide/makeinfo top

`'@top'`
=====

The `'@top'` command is a special sectioning command that you use only after an `'@node Top'` line at the beginning of a Texinfo file. The `'@top'` command tells the `'makeinfo'` formatter which node is the `'Top'` node. It has the same typesetting effect as `'@unnumbered'` (see `'@unnumbered'`). For detailed information, see See The `'@top'` Command.

1.57 texi.guide/chapter

`'@chapter'`
=====

`'@chapter'` identifies a chapter in the document. Write the command at the beginning of a line and follow it on the same line by the title of the chapter.

For example, this chapter in this manual is entitled "Chapter Structuring"; the `@chapter` line looks like this:

```
@chapter Chapter Structuring
```

In TeX, the `@chapter` command creates a chapter in the document, specifying the chapter title. The chapter is numbered automatically.

In Info, the `@chapter` command causes the title to appear on a line by itself, with a line of asterisks inserted underneath. Thus, in Info, the above example produces the following output:

```
Chapter Structuring
*****
```

1.58 texi.guide/unnumbered & appendix

```
@unnumbered', '@appendix'
=====
```

Use the `@unnumbered` command to create a chapter that appears in a printed manual without chapter numbers of any kind. Use the `@appendix` command to create an appendix in a printed manual that is labelled by letter instead of by number.

For Info file output, the `@unnumbered` and `@appendix` commands are equivalent to `@chapter`: the title is printed on a line by itself with a line of asterisks underneath. (See `@chapter`.)

To create an appendix or an unnumbered chapter, write an `@appendix` or `@unnumbered` command at the beginning of a line and follow it on the same line by the title, as you would if you were creating a chapter.

1.59 texi.guide/majorheading & chapheading

```
@majorheading', '@chapheading'
=====
```

The `@majorheading` and `@chapheading` commands put chapter-like headings in the body of a document.

However, neither command causes TeX to produce a numbered heading or an entry in the table of contents; and neither command causes TeX to start a new page in a printed manual.

In TeX, an `@majorheading` command generates a larger vertical whitespace before the heading than an `@chapheading` command but is otherwise the same.

In Info, the `@majorheading` and `@chapheading` commands are

equivalent to `'@chapter'`: the title is printed on a line by itself with a line of asterisks underneath. (See `'@chapter'`.)

1.60 texi.guide/section

`'@section'`
=====

In a printed manual, an `'@section'` command identifies a numbered section within a chapter. The section title appears in the table of contents. In Info, an `'@section'` command provides a title for a segment of text, underlined with `'='`.

This section is headed with an `'@section'` command and looks like this in the Texinfo file:

```
@section @code{@@section}
```

To create a section, write the `'@section'` command at the beginning of a line and follow it on the same line by the section title.

Thus,

```
@section This is a section
```

produces

```
This is a section  
=====
```

in Info.

1.61 texi.guide/unnumberedsec appendixsec heading

`'@unnumberedsec'`, `'@appendixsec'`, `'@heading'`
=====

The `'@unnumberedsec'`, `'@appendixsec'`, and `'@heading'` commands are, respectively, the unnumbered, appendix-like, and heading-like equivalents of the `'@section'` command. (See `'@section'`.)

`'@unnumberedsec'`

The `'@unnumberedsec'` command may be used within an unnumbered chapter or within a regular chapter or appendix to provide an unnumbered section.

`'@appendixsec'`

`'@appendixsection'`

`'@appendixsection'` is a longer spelling of the `'@appendixsec'` command; the two are synonymous.

Conventionally, the `'@appendixsec'` or `'@appendixsection'` command is used only within appendices.

`'@heading'`

You may use the `'@heading'` command anywhere you wish for a section-style heading that will not appear in the table of contents.

1.62 texi.guide/subsection

The `'@subsection'` Command

=====

Subsections are to sections as sections are to chapters. (See `'@section'`.) In Info, subsection titles are underlined with `'-'`. For example,

```
@subsection This is a subsection
```

produces

```
This is a subsection
-----
```

In a printed manual, subsections are listed in the table of contents and are numbered three levels deep.

1.63 texi.guide/unnumberedsubsec appendixsubsec subheading

The `'@subsection'-like` Commands

=====

The `'@unnumberedsubsec'`, `'@appendixsubsec'`, and `'@subheading'` commands are, respectively, the unnumbered, appendix-like, and heading-like equivalents of the `'@subsection'` command. (See `'@subsection'`.)

In Info, the `'@subsection'-like` commands generate a title underlined with hyphens. In a printed manual, an `'@subheading'` command produces a heading like that of a subsection except that it is not numbered and does not appear in the table of contents. Similarly, an `'@unnumberedsubsec'` command produces an unnumbered heading like that of a subsection and an `'@appendixsubsec'` command produces a subsection-like heading labelled with a letter and numbers; both of these commands produce headings that appear in the table of contents.

1.64 texi.guide/subsubsection

The `'subsub'` Commands

=====

The fourth and lowest level sectioning commands in Texinfo are the `'subsub'` commands. They are:

`'@subsubsection'`

Subsubsections are to subsections as subsections are to sections. (See `'@subsection'`.) In a printed manual, subsubsection titles appear in the table of contents and are numbered four levels deep.

`'@unnumberedsubsubsec'`

Unnumbered subsubsection titles appear in the table of contents of a printed manual, but lack numbers. Otherwise, unnumbered subsubsections are the same as subsubsections. In Info, unnumbered subsubsections look exactly like ordinary subsubsections.

`'@appendixsubsubsec'`

Conventionally, appendix commands are used only for appendices and are lettered and numbered appropriately in a printed manual. They also appear in the table of contents. In Info, appendix subsubsections look exactly like ordinary subsubsections.

`'@subsubheading'`

The `'@subsubheading'` command may be used anywhere that you need a small heading that will not appear in the table of contents. In Info, subsubheadings look exactly like ordinary subsubsection headings.

In Info, `'subsub'` titles are underlined with periods. For example,

```
@subsubsection This is a subsubsection
```

produces

```
This is a subsubsection
.....
```

1.65 texi.guide/Nodes

Nodes

"Nodes" are the primary segments of a Texinfo file. They do not themselves impose a hierarchic or any other kind of structure on a file. Nodes contain "node pointers" that name other nodes, and can contain "menus" which are lists of nodes. In Info, the movement commands can carry you to a pointed-to node or to a node listed in a menu. Node pointers and menus provide structure for Info files just as chapters, sections, subsections, and the like, provide structure for printed books.

Two Paths	Different commands to structure Info output and printed output.
Node Menu Illustration	A diagram, and sample nodes and menus.
node	How to write a node, in detail.
makeinfo Pointer Creation	How to create node pointers with 'makeinfo'.

1.66 texi.guide/Two Paths

Two Paths
=====

The node and menu commands and the chapter structuring commands are independent of each other:

- * In Info, node and menu commands provide structure. The chapter structuring commands generate headings with different kinds of underlining--asterisks for chapters, hyphens for sections, and so on; they do nothing else.
- * In TeX, the chapter structuring commands generate chapter and section numbers and tables of contents. The node and menu commands provide information for cross references; they do nothing else.

You can use node pointers and menus to structure an Info file any way you want; and you can write a Texinfo file so that its Info output has a different structure than its printed output. However, most Texinfo files are written such that the structure for the Info output corresponds to the structure for the printed output. It is not convenient to do otherwise.

Generally, printed output is structured in a tree-like hierarchy in which the chapters are the major limbs from which the sections branch out. Similarly, node pointers and menus are organized to create a matching structure in the Info output.

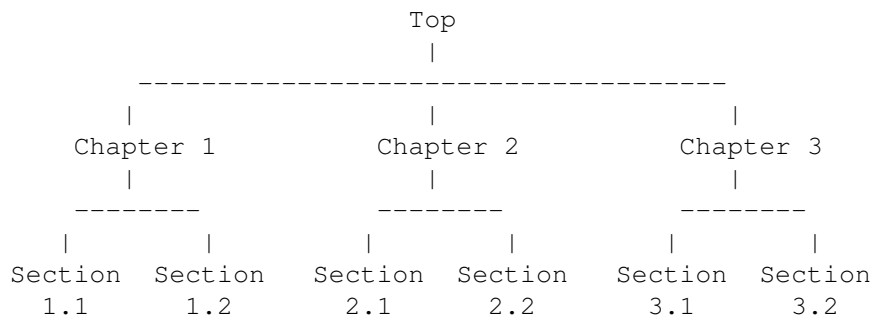
1.67 texi.guide/Node Menu Illustration

Node and Menu Illustration
=====

Here is a copy of the diagram shown earlier that illustrates a Texinfo file with three chapters, each of which contains two sections.

Note that the "root" is at the top of the diagram and the "leaves" are at the bottom. This is how such a diagram is drawn conventionally; it illustrates an upside-down tree. For this reason, the root node is called the 'Top' node, and 'Up' node pointers carry you closer to the

root.



Write the beginning of the node for Chapter 2 like this:

```
@node      Chapter 2, Chapter 3, Chapter 1, top
@comment   node-name, next,      previous, up
```

This '@node' line says that the name of this node is "Chapter 2", the name of the 'Next' node is "Chapter 3", the name of the 'Previous' node is "Chapter 1", and the name of the 'Up' node is "Top".

Please Note:* 'Next' refers to the next node at the same hierarchical level in the manual, not necessarily to the next node within the Texinfo file. In the Texinfo file, the subsequent node may be at a lower level--a section-level node may follow a chapter-level node, and a subsection-level node may follow a section-level node. 'Next' and 'Previous' refer to nodes at the **same hierarchical level. (The 'Top' node contains the exception to this rule. Since the 'Top' node is the only node at that level, 'Next' refers to the first following node, which is almost always a chapter or chapter-level node.)

To go to Sections 2.1 and 2.2 using Info, you need a menu inside Chapter 2. (See Menus.) You would write the menu just before the beginning of Section 2.1, like this:

```
@menu
* Sect. 2.1::      Description of this section.
* Sect. 2.2::
@end menu
```

Write the node for Sect. 2.1 like this:

```
@node      Sect. 2.1, Sect. 2.2, Chapter 2, Chapter 2
@comment   node-name, next,      previous, up
```

In Info format, the 'Next' and 'Previous' pointers of a node usually lead to other nodes at the same level--from chapter to chapter or from section to section (sometimes, as shown, the 'Previous' pointer points up); an 'Up' pointer usually leads to a node at the level above (closer to the 'Top' node); and a 'Menu' leads to nodes at a level below (closer to 'leaves'). (A cross reference can point to a node at any level; see See Cross References.)

Usually, an '@node' command and a chapter structuring command are used in sequence, along with indexing commands. (You may follow the

'@node' line with a comment line that reminds you which pointer is which.)

Here is the beginning of the chapter in this manual called "Ending a Texinfo File". This shows an '@node' line followed by a comment line, an '@chapter' line, and then by indexing lines.

```
@node      Ending a File, Structuring, Beginning a File, Top
@comment node-name,      next,      previous,      up
@chapter Ending a Texinfo File
@cindex Ending a Texinfo file
@cindex Texinfo file ending
@cindex File ending
```

1.68 texi.guide/node

The '@node' Command

=====

A "node" is a segment of text that begins at an '@node' command and continues until the next '@node' command. The definition of node is different from that for chapter or section. A chapter may contain sections and a section may contain subsections; but a node cannot contain subnodes; the text of a node continues only until the next '@node' command in the file. A node usually contains only one chapter structuring command, the one that follows the '@node' line. On the other hand, in printed output nodes are used only for cross references, so a chapter or section may contain any number of nodes. Indeed, a chapter usually contains several nodes, one for each section, subsection, and subsubsection.

To create a node, write an '@node' command at the beginning of a line, and follow it with four arguments, separated by commas, on the rest of the same line. These arguments are the name of the node, and the names of the 'Next', 'Previous', and 'Up' pointers, in that order. You may insert spaces before each pointer if you wish; the spaces are ignored. You must write the name of the node, and the names of the 'Next', 'Previous', and 'Up' pointers, all on the same line. Otherwise, the formatters fail. (See info, for more information about nodes in Info.)

Usually, you write one of the chapter-structuring command lines immediately after an '@node' line--for example, an '@section' or '@subsection' line. (See Types of Structuring Command.)

Please note: The GNU Emacs Texinfo mode updating commands work only with Texinfo files in which '@node' lines are followed by chapter structuring lines. See Updating Requirements.

TeX uses '@node' lines to identify the names to use for cross references. For this reason, you must write '@node' lines in a Texinfo file that you intend to format for printing, even if you do not intend to format it for Info. (Cross references, such as the one at the end of this sentence, are made with '@xref' and its related commands; see

See Cross References.)

Node Names	How to choose node and pointer names.
Writing a Node	How to write an '@node' line.
Node Line Tips	Keep names short.
Node Line Requirements	Keep names unique, without @-commands.
First Node	How to write a 'Top' node.
makeinfo top command	How to use the '@top' command.
Top Node Summary	Write a brief description for readers.

1.69 texi.guide/Node Names

Choosing Node and Pointer Names

The name of a node identifies the node. The pointers enable you to reach other nodes and consist of the names of those nodes.

Normally, a node's 'Up' pointer contains the name of the node whose menu mentions that node. The node's 'Next' pointer contains the name of the node that follows that node in that menu and its 'Previous' pointer contains the name of the node that precedes it in that menu. When a node's 'Previous' node is the same as its 'Up' node, both node pointers name the same node.

Usually, the first node of a Texinfo file is the 'Top' node, and its 'Up' and 'Previous' pointers point to the 'dir' file, which contains the main menu for all of Info.

The 'Top' node itself contains the main or master menu for the manual. Also, it is helpful to include a brief description of the manual in the 'Top' node. See First Node, for information on how to write the first node of a Texinfo file.

1.70 texi.guide/Writing a Node

How to Write an '@node' Line

The easiest way to write an '@node' line is to write '@node' at the beginning of a line and then the name of the node, like this:

```
@node NODE-NAME
```

If you are using GNU Emacs, you can use the update node commands provided by Texinfo mode to insert the names of the pointers; or you can leave the pointers out of the Texinfo file and let 'makeinfo' insert node pointers into the Info file it creates. (See Texinfo Mode, and See makeinfo Pointer Creation.)

Alternatively, you can insert the 'Next', 'Previous', and 'Up' pointers yourself. If you do this, you may find it helpful to use the Texinfo mode keyboard command 'C-c C-c n'. This command inserts '@node' and a comment line listing the names of the pointers in their proper order. The comment line helps you keep track of which arguments are for which pointers. This comment line is especially useful if you are not familiar with Texinfo.

The template for a node line with 'Next', 'Previous', and 'Up' pointers looks like this:

```
@node NODE-NAME, NEXT, PREVIOUS, UP
```

If you wish, you can ignore '@node' lines altogether in your first draft and then use the 'texinfo-insert-node-lines' command to create '@node' lines for you. However, we do not recommend this practice. It is better to name the node itself at the same time that you write a segment so you can easily make cross references. A large number of cross references are an especially important feature of a good Info file.

After you have inserted an '@node' line, you should immediately write an @-command for the chapter or section and insert its name. Next (and this is important!), put in several index entries. Usually, you will find at least two and often as many as four or five ways of referring to the node in the index. Use them all. This will make it much easier for people to find the node.

1.71 texi.guide/Node Line Tips

'@node' Line Tips

Here are three suggestions:

- * Try to pick node names that are informative but short.

In the Info file, the file name, node name, and pointer names are all inserted on one line, which may run into the right edge of the window. (This does not cause a problem with Info, but is ugly.)

- * Try to pick node names that differ from each other near the beginnings of their names. This way, it is easy to use automatic name completion in Info.
 - * By convention, node names are capitalized just as they would be for section or chapter titles--initial and significant words are capitalized; others are not.
-

1.72 texi.guide/Node Line Requirements

``@node'` Line Requirements

Here are several requirements for ``@node'` lines:

- * All the node names for a single Info file must be unique.

Duplicates confuse the Info movement commands. This means, for example, that if you end every chapter with a summary, you must name each summary node differently. You cannot just call each one "Summary". You may, however, duplicate the titles of chapters, sections, and the like. Thus you can end each chapter in a book with a section called "Summary", so long as the node names for those sections are all different.

- * A pointer name must be the name of a node.

The node to which a pointer points may come before or after the node containing the pointer.

- * You cannot use any of the Texinfo `@`-commands in a node name; `@`-commands confuse Info.

Thus, the beginning of the section called ``@chapter'` looks like this:

```
@node chapter, unnumbered & appendix, makeinfo top, Structuring
@comment node-name, next, previous, up
@section @code{@@chapter}
@findex chapter
```

- * You cannot use commas, colons, or apostrophes within a node name; these confuse TeX or the Info formatters.

For example, the following is a section title:

```
@code{@@unnumberedsec}, @code{@@appendixsec}, @code{@@heading}
```

The corresponding node name is:

```
unnumberedsec appendixsec heading
```

- * Case is significant.

1.73 texi.guide/First Node

The First Node

The first node of a Texinfo file is the `'Top'` node, except in an included file (see Include Files).

The 'Top' node (which must be named 'top' or 'Top') should have as its 'Up' and 'Previous' nodes the name of a node in another file, where there is a menu that leads to this file. Specify the file name in parentheses. If the file is to be installed directly in the Info directory file, use '(dir)' as the parent of the 'Top' node; this is short for '(dir)top', and specifies the 'Top' node in the 'dir' file, which contains the main menu for Info. For example, the '@node Top' line of this manual looks like this:

```
@node Top, Overview, (dir), (dir)
```

(You may use the Texinfo updating commands or the 'makeinfo' utility to insert these 'Next' and '(dir)' pointers automatically.)

See Install an Info File, for more information about installing an Info file in the 'info' directory.

The 'Top' node contains the main or master menu for the document.

1.74 texi.guide/makeinfo top command

The '@top' Sectioning Command

A special sectioning command, '@top', has been created for use with the '@node Top' line. The '@top' sectioning command tells 'makeinfo' that it marks the 'Top' node in the file. It provides the information that 'makeinfo' needs to insert node pointers automatically. Write the '@top' command at the beginning of the line immediately following the '@node Top' line. Write the title on the remaining part of the same line as the '@top' command.

In Info, the '@top' sectioning command causes the title to appear on a line by itself, with a line of asterisks inserted underneath.

In TeX and 'texinfo-format-buffer', the '@top' sectioning command is merely a synonym for '@unnumbered'. Neither of these formatters require an '@top' command, and do nothing special with it. You can use '@chapter' or '@unnumbered' after the '@node Top' line when you use these formatters. Also, you can use '@chapter' or '@unnumbered' when you use the Texinfo updating commands to create or update pointers and menus.

Whatever sectioning command follows an '@node Top' line, whether it be '@top' or '@chapter', the '@node Top' line and the immediately following line and any additional text must be enclosed between '@ifinfo' and '@end ifinfo' commands. (See Conditionals.) This prevents the title and the accompanying text from appearing in printed output. Write the '@ifinfo' command before the '@node' line and write the '@end ifinfo' command after the '@top' or other sectioning command and after any additional text. (You can write the '@end ifinfo' command after the '@end menu' command if you like.)

1.75 texi.guide/Top Node Summary

The 'Top' Node Summary

You can help readers by writing a summary in the 'Top' node, after the '@top' line, before the main or master menu. The summary should briefly describe the Info file. You should also write the version number of the program to which the manual applies in this section. This helps the reader keep track of which manual is for which version of the program. If the manual changes more frequently than the program or is independent of it, you should also include an edition number for the manual. (The title page should also contain this information: see See '@titlepage'.)

Put the whole of the 'Top' node, including the '@top' sectioning command line if you have one, between '@ifinfo' and '@end ifinfo' so none of the text appears in the printed output (see Conditionally Visible Text). (You may want to repeat the brief description from the 'Top' node within '@iftex' ... '@end iftex' at the beginning of the first chapter, for those who read the printed manual.)

1.76 texi.guide/makeinfo Pointer Creation

Creating Pointers with 'makeinfo'

=====

The 'makeinfo' program has a feature for automatically creating node pointers for a hierarchically organized file that lacks them.

When you take advantage of this feature, you do not need to write the 'Next', 'Previous', and 'Up' pointers after the name of a node. However, you must write a sectioning command, such as '@chapter' or '@section', on the line immediately following each truncated '@node' line. You cannot write a comment line after a node line; the section line must follow it immediately.

In addition, you must follow the 'Top' '@node' line with a line beginning with '@top' to mark the 'Top' node in the file. See '@top'.

Finally, you must write the name of each node (except for the 'Top' node) in a menu that is one or more hierarchical levels above the node's hierarchical level.

This node pointer insertion feature in 'makeinfo' is an alternative to the menu and pointer creation and update commands in Texinfo mode. (See Updating Nodes and Menus.) It is especially helpful to people who do not use GNU Emacs for writing Texinfo documents.

1.77 texi.guide/Menus

Menus

"Menus" contain pointers to subordinate nodes.(1) In Info, you use menus to go to such nodes. Menus have no effect in printed manuals and do not appear in them.

By convention, a menu is put at the end of a node since a reader who uses the menu may not see text that follows it.

A node that has a menu should *not* contain much text. If you have a lot of text and a menu, move most of the text into a new subnode--all but a few lines.

Menu Location	Put a menu in a short node.
Writing a Menu	What is a menu?
Menu Parts	A menu entry has three parts.
Less Cluttered Menu Entry	Two part menu entry.
Menu Example	Two and three part menu entries.
Other Info Files	How to refer to a different Info file.

----- Footnotes -----

(1) Menus can carry you to any node, regardless of the hierarchical structure; even to nodes in a different Info file. However, the GNU Emacs Texinfo mode updating commands work only to create menus of subordinate nodes. Conventionally, cross references are used to refer to other nodes.

1.78 texi.guide/Menu Location

Menus Need Short Nodes

=====

A reader can easily see a menu that is close to the beginning of the node. The node should be short. As a practical matter, you should locate a menu within 20 lines of the beginning of the node. Otherwise, a reader with a terminal that displays only a few lines may miss the menu and its associated text.

The short text before a menu may look awkward in a printed manual. To avoid this, you can write a menu near the beginning of its node and follow the menu by an '@node' line, and then an '@heading' line located within '@ifinfo' and '@end ifinfo'. This way, the menu, '@node' line, and title appear only in the Info file, not the printed document.

For example, the preceding two paragraphs follow an Info-only menu, '@node' line, and heading, and look like this:

```
@menu
```

```

* Menu Location::          Put a menu in a short node.
* Writing a Menu::         What is a menu?
* Menu Parts::             A menu entry has three parts.
* Less Cluttered Menu Entry:: Two part menu entry.
* Menu Example::          Two and three part entries.
* Other Info Files::       How to refer to a different
                           Info file.

@end menu

@node Menu Location, Writing a Menu, , Menus
@ifinfo
@heading Menus Need Short Nodes
@end ifinfo

```

The Texinfo file for this document contains more than a dozen examples of this procedure. One is at the beginning of this chapter; another is at the beginning of the "Cross References" chapter.

1.79 texi.guide/Writing a Menu

Writing a Menu
=====

A menu consists of an '@menu' command on a line by itself followed by menu entry lines or menu comment lines and then by an '@end menu' command on a line by itself.

A menu looks like this:

```

@menu
Larger Units of Text

* Files::          All about handling files.
* Multiples: Buffers. Multiple buffers; editing
                   several files at once.

@end menu

```

In a menu, every line that begins with an '@*' is a "menu entry". (Note the space after the asterisk.) A line that does not start with an '@*' may also appear in a menu. Such a line is not a menu entry but is a menu comment line that appears in the Info file. In the example above, the line 'Larger Units of Text' is a menu comment line; the two lines starting with '@*' are menu entries.

1.80 texi.guide/Menu Parts

The Parts of a Menu
=====

A menu entry has three parts, only the second of which is required:

1. The menu entry name.
2. The name of the node (required).
3. A description of the item.

The template for a menu entry looks like this:

```
* MENU-ENTRY-NAME: NODE-NAME.    DESCRIPTION
```

Follow the menu entry name with a single colon and follow the node name with tab, comma, period, or newline.

In Info, a user selects a node with the 'm' ('Info-menu') command. The menu entry name is what the user types after the 'm' command.

The third part of a menu entry is a descriptive phrase or sentence. Menu entry names and node names are often short; the description explains to the reader what the node is about. The description, which is optional, can spread over two or more lines. A useful description complements the node name rather than repeats it.

1.81 texi.guide/Less Cluttered Menu Entry

Less Cluttered Menu Entry
=====

When the menu entry name and node name are the same, you can write the name immediately after the asterisk and space at the beginning of the line and follow the name with two colons.

For example, write

```
* Name::                                DESCRIPTION
```

instead of

```
* Name: Name.                            DESCRIPTION
```

You should use the node name for the menu entry name whenever possible, since it reduces visual clutter in the menu.

1.82 texi.guide/Menu Example

A Menu Example
=====

A menu looks like this in Texinfo:

```
@menu
* menu entry name: Node name.    A short description.
* Node name::                  This form is preferred.
@end menu
```

This produces:

```
* menu:

* menu entry name: Node name.    A short description.
* Node name::                  This form is preferred.
```

Here is an example as you might see it in a Texinfo file:

```
@menu
Larger Units of Text

* Files::                      All about handling files.
* Multiples: Buffers.          Multiple buffers; editing
                               several files at once.
@end menu
```

This produces:

```
* menu:
Larger Units of Text

* Files::                      All about handling files.
* Multiples: Buffers.          Multiple buffers; editing
                               several files at once.
```

In this example, the menu has two entries. 'Files' is both a menu entry name and the name of the node referred to by that name. 'Multiples' is the menu entry name; it refers to the node named 'Buffers'. The line 'Larger Units of Text' is a comment; it appears in the menu, but is not an entry.

Since no file name is specified with either 'Files' or 'Buffers', they must be the names of nodes in the same Info file (see Referring to Other Info Files).

1.83 texi.guide/Other Info Files

Referring to Other Info Files

=====

You can create a menu entry that enables a reader in Info to go to a node in another Info file by writing the file name in parentheses just before the node name. In this case, you should use the three-part menu entry format, which saves the reader from having to type the file name.

The format looks like this:

```
@menu
```

```
* FIRST-ENTRY-NAME: (FILENAME) NODENAME.      DESCRIPTION
* SECOND-ENTRY-NAME: (FILENAME) SECOND-NODE. DESCRIPTION
@end menu
```

For example, to refer directly to the 'Outlining' and 'Rebinding' nodes in the 'Emacs Manual', you would write a menu like this:

```
@menu
* Outlining: (emacs) Outline Mode. The major mode for
                                editing outlines.
* Rebinding: (emacs) Rebinding.   How to redefine the
                                meaning of a key.
@end menu
```

If you do not list the node name, but only name the file, then Info presumes that you are referring to the 'Top' node.

The 'dir' file that contains the main menu for Info has menu entries that list only file names. These take you directly to the 'Top' nodes of each Info document. (See Install an Info File.)

For example:

```
* Info: (info).      Documentation browsing system.
* Emacs: (emacs).    The extensible, self-documenting
                    text editor.
```

(The 'dir' top level directory for the Info system is an Info file, not a Texinfo file, but a menu entry looks the same in both types of file.)

Note that the GNU Emacs Texinfo mode menu updating commands only work with nodes within the current buffer, so you cannot use them to create menus that refer to other files. You must write such menus by hand.

1.84 texi.guide/Cross References

Cross References

"Cross references" are used to refer the reader to other parts of the same or different Texinfo files. In Texinfo, nodes are the places to which cross references can refer.

References	What cross references are for.
Cross Reference Commands	A summary of the different commands.
Cross Reference Parts	A cross reference has several parts.
xref	Begin a reference with 'See' ...
Top Node Naming	How to refer to the beginning of another file.
ref	A reference for the last part of a sentence.
pxref	How to write a parenthetical cross reference.
inforef	How to refer to an Info-only file.

1.85 texi.guide/References

What References Are For

=====

Often, but not always, a printed document should be designed so that it can be read sequentially. People tire of flipping back and forth to find information that should be presented to them as they need it.

However, in any document, some information will be too detailed for the current context, or incidental to it; use cross references to provide access to such information. Also, an on-line help system or a reference manual is not like a novel; few read such documents in sequence from beginning to end. Instead, people look up what they need. For this reason, such creations should contain many cross references to help readers find other information that they may not have read.

In a printed manual, a cross reference results in a page reference, unless it is to another manual altogether, in which case the cross reference names that manual.

In Info, a cross reference results in an entry that you can follow using the Info 'f' command. (See Some advanced Info commands.)

The various cross reference commands use nodes to define cross reference locations. This is evident in Info, in which a cross reference takes you to the specified node. TeX also uses nodes to define cross reference locations, but the action is less obvious. When TeX generates a DVI file, it records nodes' page numbers and uses the page numbers in making references. Thus, if you are writing a manual that will only be printed, and will not be used on-line, you must nonetheless write '@node' lines to name the places to which you make cross references.

1.86 texi.guide/Cross Reference Commands

Different Cross Reference Commands

=====

There are four different cross reference commands:

'@xref'

Used to start a sentence in the printed manual saying 'See ...' or an entry in the Info file saying '*Note ...'.

'@ref'

Used within or, more often, at the end of a sentence; same as '@xref' for Info; produces just the reference in the printed manual without a preceding 'See'.

'@pxref'

Used within parentheses to make a reference that suits both an Info

file and a printed book. Starts with a lower case 'see' within the printed manual. ('p' is for 'parenthesis'.)

`'@inforef'`

Used to make a reference to an Info file for which there is no printed manual.

(The `'@cite'` command is used to make references to books and manuals for which there is no corresponding Info file and, therefore, no node to which to point. See `'@cite'`.)

1.87 texi.guide/Cross Reference Parts

Parts of a Cross Reference

=====

A cross reference command requires only one argument, which is the name of the node to which it refers. But a cross reference command may contain up to four additional arguments. By using these arguments, you can provide a cross reference name for Info, a topic description or section title for the printed output, the name of a different Info file, and the name of a different printed manual.

Here is a simple cross reference example:

```
@xref{Node name}.
```

which produces

```
*Note Node name::.
```

and

```
See Section NNN [Node name], page PPP.
```

Here is an example of a full five-part cross reference:

```
@xref{Node name, Cross Reference Name, Particular Topic,
info-file-name, A Printed Manual}, for details.
```

which produces

```
*Note Cross Reference Name: (info-file-name)Node name,
for details.
```

in Info and

```
See section "Particular Topic" in A Printed Manual, for details.
```

in a printed book.

The five possible arguments for a cross reference are:

1. The node name (required). This is the node to which the cross

reference takes you. In a printed document, the location of the node provides the page reference only for references within the same document.

2. The cross reference name for the Info reference, if it is to be different from the node name. If you include this argument, it argument becomes the first part of the cross reference. It is usually omitted.
3. A topic description or section name. Often, this is the title of the section. This is used as the name of the reference in the printed manual. If omitted, the node name is used.
4. The name of the Info file in which the reference is located, if it is different from the current file.
5. The name of a printed manual from a different Texinfo file.

The template for a full five argument cross reference looks like this:

```
@xref{NODE-NAME, CROSS-REFERENCE-NAME, TITLE-OR-TOPIC,
INFO-FILE-NAME, PRINTED-MANUAL-TITLE}.
```

Cross references with one, two, three, four, and five arguments are described separately following the description of '@xref'.

Write a node name in a cross reference in exactly the same way as in the '@node' line, including the same capitalization; otherwise, the formatters may not find the reference.

You can write cross reference commands within a paragraph, but note how Info and TeX format the output of each of the various commands: write '@xref' at the beginning of a sentence; write '@pxref' only within parentheses, and so on.

1.88 texi.guide/xref

```
'@xref'
=====
```

The '@xref' command generates a cross reference for the beginning of a sentence. The Info formatting commands convert it into an Info cross reference, which the Info 'f' command can use to bring you directly to another node. The TeX typesetting commands convert it into a page reference, or a reference to another book or manual.

Reference Syntax	What a reference looks like and requires.
One Argument	'@xref' with one argument.
Two Arguments	'@xref' with two arguments.
Three Arguments	'@xref' with three arguments.
Four and Five Arguments	'@xref' with four and five arguments.

1.89 texi.guide/Reference Syntax

What a Reference Looks Like and Requires

Most often, an Info cross reference looks like this:

```
*Note NODE-NAME::.
```

or like this

```
*Note CROSS-REFERENCE-NAME: NODE-NAME.
```

In TeX, a cross reference looks like this:

```
See Section SECTION-NUMBER [NODE-NAME], page PAGE.
```

or like this

```
See Section SECTION-NUMBER [TITLE-OR-TOPIC], page PAGE.
```

The `'@xref'` command does not generate a period or comma to end the cross reference in either the Info file or the printed output. You must write that period or comma yourself; otherwise, Info will not recognize the end of the reference. (The `'@pxref'` command works differently. See `'@pxref'`.)

Please note: A period or comma **must** follow the closing brace of an `'@xref'`. It is required to terminate the cross reference. This period or comma will appear in the output, both in the Info file and in the printed manual.

`'@xref'` must refer to an Info node by name. Use `'@node'` to define the node (see Writing a Node).

`'@xref'` is followed by several arguments inside braces, separated by commas. Whitespace before and after these commas is ignored.

A cross reference requires only the name of a node; but it may contain up to four additional arguments. Each of these variations produces a cross reference that looks somewhat different.

Please note: Commas separate arguments in a cross reference; avoid including them in the title or other part lest the formatters mistake them for separators.

1.90 texi.guide/One Argument

`'@xref'` with One Argument

The simplest form of `'@xref'` takes one argument, the name of another node in the same Info file. The Info formatters produce output that the Info readers can use to jump to the reference; TeX produces output that specifies the page and section number for you.

For example,

```
@xref{Tropical Storms}.
```

produces

```
*Note Tropical Storms::.
```

and

```
See Section 3.1 [Tropical Storms], page 24.
```

(Note that in the preceding example the closing brace is followed by a period.)

You can write a clause after the cross reference, like this:

```
@xref{Tropical Storms}, for more info.
```

which produces

```
*Note Tropical Storms::, for more info.
```

```
See Section 3.1 [Tropical Storms], page 24, for more info.
```

(Note that in the preceding example the closing brace is followed by a comma, and then by the clause, which is followed by a period.)

1.91 texi.guide/Two Arguments

`'@xref'` with Two Arguments

With two arguments, the second is used as the name of the Info cross reference, while the first is still the name of the node to which the cross reference points.

The template is like this:

```
@xref{NODE-NAME, CROSS-REFERENCE-NAME}.
```

For example,

```
@xref{Electrical Effects, Lightning}.
```

produces:

```
*Note Lightning: Electrical Effects.
```

and

```
See Section 5.2 [Electrical Effects], page 57.
```

(Note that in the preceding example the closing brace is followed by a period; and that the node name is printed, not the cross reference name.)

You can write a clause after the cross reference, like this:

```
@xref{Electrical Effects, Lightning}, for more info.
```

which produces

```
*Note Lightning: Electrical Effects, for more info.
```

and

```
See Section 5.2 [Electrical Effects], page 57, for more info.
```

(Note that in the preceding example the closing brace is followed by a comma, and then by the clause, which is followed by a period.)

1.92 texi.guide/Three Arguments

'@xref' with Three Arguments

A third argument replaces the node name in the TeX output. The third argument should be the name of the section in the printed output, or else state the topic discussed by that section. Often, you will want to use initial upper case letters so it will be easier to read when the reference is printed. Use a third argument when the node name is unsuitable because of syntax or meaning.

Remember to avoid placing a comma within the title or topic section of a cross reference, or within any other section. The formatters divide cross references into arguments according to the commas; a comma within a title or other section will divide it into two arguments. In a reference, you need to write a title such as "Clouds, Mist, and Fog" without the commas.

Also, remember to write a comma or period after the closing brace of a '@xref' to terminate the cross reference. In the following examples, a clause follows a terminating comma.

The template is like this:

```
@xref{NODE-NAME, CROSS-REFERENCE-NAME, TITLE-OR-TOPIC}.
```

For example,

```
@xref{Electrical Effects, Lightning, Thunder and Lightning},
for details.
```

produces

```
*Note Lightning: Electrical Effects, for details.
```

and

```
See Section 5.2 [Thunder and Lightning], page 57, for details.
```

If a third argument is given and the second one is empty, then the third argument serves both. (Note how two commas, side by side, mark the empty second argument.)

```
@xref{Electrical Effects, , Thunder and Lightning},
for details.
```

produces

```
*Note Thunder and Lightning: Electrical Effects, for details.
```

and

```
See Section 5.2 [Thunder and Lightning], page 57, for details.
```

As a practical matter, it is often best to write cross references with just the first argument if the node name and the section title are the same, and with the first and third arguments if the node name and title are different.

Here are several examples from 'The GAWK Manual':

```
@xref{Sample Program}.
@xref{Glossary}.
@xref{Case-sensitivity, ,Case-sensitivity in Matching}.
@xref{Close Output, , Closing Output Files and Pipes},
    for more information.
@xref{Regexp, , Regular Expressions as Patterns}.
```

1.93 texi.guide/Four and Five Arguments

'@xref' with Four and Five Arguments

In a cross reference, a fourth argument specifies the name of another Info file, different from the file in which the reference appears, and a fifth argument specifies its title as a printed manual.

Remember that a comma or period must follow the closing brace of an '@xref' command to terminate the cross reference. In the following examples, a clause follows a terminating comma.

The template is:

```
@xref{NODE-NAME, CROSS-REFERENCE-NAME, TITLE-OR-TOPIC,
INFO-FILE-NAME, PRINTED-MANUAL-TITLE}.
```

For example,

```
@xref{Electrical Effects, Lightning, Thunder and Lightning,
weather, An Introduction to Meteorology}, for details.
```

produces

```
*Note Lightning: (weather)Electrical Effects, for details.
```

The name of the Info file is enclosed in parentheses and precedes the name of the node.

In a printed manual, the reference looks like this:

```
See section "Thunder and Lightning" in An Introduction to
Meteorology, for details.
```

The title of the printed manual is typeset in italics; and the reference lacks a page number since TeX cannot know to which page a reference refers when that reference is to another manual.

Often, you will leave out the second argument when you use the long version of '@xref'. In this case, the third argument, the topic description, will be used as the cross reference name in Info.

The template looks like this:

```
@xref{NODE-NAME, , TITLE-OR-TOPIC, INFO-FILE-NAME,
PRINTED-MANUAL-TITLE}, for details.
```

which produces

```
*Note TITLE-OR-TOPIC: (INFO-FILE-NAME)NODE-NAME, for details.
```

and

```
See section TITLE-OR-TOPIC in PRINTED-MANUAL-TITLE, for details.
```

For example,

```
@xref{Electrical Effects, , Thunder and Lightning,
weather, An Introduction to Meteorology}, for details.
```

produces

```
*Note Thunder and Lightning: (weather)Electrical Effects,
for details.
```

and

```
See section "Thunder and Lightning" in An Introduction to
Meteorology, for details.
```

On rare occasions, you may want to refer to another Info file that is within a single printed manual--when multiple Texinfo files are incorporated into the same TeX run but make separate Info files. In this case, you need to specify only the fourth argument, and not the fifth.

1.94 texi.guide/Top Node Naming

Naming a 'Top' Node

=====

In a cross reference, you must always name a node. This means that in order to refer to a whole manual, you must identify the 'Top' node by writing it as the first argument to the '@xref' command. (This is different from the way you write a menu entry; see See Referring to Other Info Files.) At the same time, to provide a meaningful section topic or title in the printed cross reference (instead of the word 'Top'), you must write an appropriate entry for the third argument to the '@xref' command.

Thus, to make a cross reference to 'The GNU Make Manual', write:

```
@xref{Top, , Overview, make, The GNU Make Manual}.
```

which produces

```
*Note Overview: (make)Top.
```

and

```
See section "Overview" in The GNU Make Manual.
```

In this example, 'Top' is the name of the first node, and 'Overview' is the name of the first section of the manual.

1.95 texi.guide/ref

'@ref'

=====

'@ref' is nearly the same as '@xref' except that it does not generate a 'See' in the printed output, just the reference itself. This makes it useful as the last part of a sentence.

For example,

```
For more information, see @ref{Hurricanes}.
```

produces

For more information, see `*Note Hurricanes`.

and

For more information, see Section 8.2 [Hurricanes], page 123.

The `'@ref'` command sometimes leads writers to express themselves in a manner that is suitable for a printed manual but looks awkward in the Info format. Bear in mind that your audience will be using both the printed and the Info format.

For example,

Sea surges are described in `@ref{Hurricanes}`.

produces

Sea surges are described in Section 6.7 [Hurricanes], page 72.

in a printed document, and the following in Info:

Sea surges are described in `*Note Hurricanes::`.

`*Caution:` You *must* write a period or comma immediately after an `'@ref'` command with two or more arguments. Otherwise, Info will not find the end of the cross reference entry and its attempt to follow the cross reference will fail. As a general rule, you should write a period or comma after every `'@ref'` command. This looks best in both the printed and the Info output.

1.96 texi.guide/pxref

`'@pxref'`
=====

The parenthetical reference command, `'@pxref'`, is nearly the same as `'@xref'`, but you use it *only* inside parentheses and you do *not* type a comma or period after the command's closing brace. The command differs from `'@xref'` in two ways:

1. TeX typesets the reference for the printed manual with a lower case 'see' rather than an upper case 'See'.
2. The Info formatting commands automatically end the reference with a closing colon or period.

Because one type of formatting automatically inserts closing punctuation and the other does not, you should use `'@pxref'` *only* inside parentheses as part of another sentence. Also, you yourself should not insert punctuation after the reference, as you do with `'@xref'`.

`'@pxref'` is designed so that the output looks right and works right

between parentheses both in printed output and in an Info file. In a printed manual, a closing comma or period should not follow a cross reference within parentheses; such punctuation is wrong. But in an Info file, suitable closing punctuation must follow the cross reference so Info can recognize its end. '@pxref' spares you the need to use complicated methods to put a terminator into one form of the output and not the other.

With one argument, a parenthetical cross reference looks like this:

```
... storms cause flooding (@pxref{Hurricanes}) ...
```

which produces

```
... storms cause flooding (*Note Hurricanes::) ...
```

and

```
... storms cause flooding (see Section 6.7 [Hurricanes], page 72)
...
```

With two arguments, a parenthetical cross reference has this template:

```
... (@pxref{NODE-NAME, CROSS-REFERENCE-NAME}) ...
```

which produces

```
... (*Note CROSS-REFERENCE-NAME: NODE-NAME.) ...
```

and

```
... (see Section NNN [NODE-NAME], page PPP) ...
```

'@pxref' can be used with up to five arguments just like '@xref' (see '@xref').

Please note: Use '@pxref' only as a parenthetical reference. Do not try to use '@pxref' as a clause in a sentence. It will look bad in either the Info file, the printed output, or both.

Also, parenthetical cross references look best at the ends of sentences. Although you may write them in the middle of a sentence, that location breaks up the flow of text.

1.97 texi.guide/inforef

```
'@inforef'
=====
```

'@inforef' is used for cross references to Info files for which there are no printed manuals. Even in a printed manual, '@inforef' generates a reference directing the user to look in an Info file.

The command takes either two or three arguments, in the following order:

1. The node name.
2. The cross reference name (optional).
3. The Info file name.

Separate the arguments with commas, as with '@xref'. Also, you must terminate the reference with a comma or period after the '}', as you do with '@xref'.

The template is:

```
@inforef{NODE-NAME, CROSS-REFERENCE-NAME, INFO-FILE-NAME},
```

Thus,

```
@inforef{Expert, Advanced Info commands, info},
for more information.
```

produces

```
*Note Advanced Info commands: (info)Expert,
for more information.
```

and

```
See Info file 'info', node 'Expert', for more information.
```

Similarly,

```
@inforef{Expert, , info}, for more information.
```

produces

```
*Note (info)Expert::, for more information.
```

and

```
See Info file 'info', node 'Expert', for more information.
```

The converse of '@inforef' is '@cite', which is used to refer to printed works for which no Info form exists. See '@cite'.

1.98 texi.guide/Marking Text

Marking Words and Phrases

In Texinfo, you can mark words and phrases in a variety of ways. The Texinfo formatters use this information to determine how to highlight the text. You can specify, for example, whether a word or

phrase is a defining occurrence, a metasyntactic variable, or a symbol used in a program. Also, you can emphasize text.

Indicating	How to indicate definitions, files, etc.
Emphasis	How to emphasize text.

1.99 texi.guide/Indicating

Indicating Definitions, Commands, etc.

=====

Texinfo has commands for indicating just what kind of object a piece of text refers to. For example, metasyntactic variables are marked by '@var', and code by '@code'. Since the pieces of text are labelled by commands that tell what kind of object they are, it is easy to change the way the Texinfo formatters prepare such text. (Texinfo is an **intentional** formatting language rather than a **typesetting** formatting language.)

For example, in a printed manual, code is usually illustrated in a typewriter font; '@code' tells TeX to typeset this text in this font. But it would be easy to change the way TeX highlights code to use another font, and this change would not effect how keystroke examples are highlighted. If straight typesetting commands were used in the body of the file and you wanted to make a change, you would need to check every single occurrence to make sure that you were changing code and not something else that should not be changed.

Useful Highlighting	Highlighting provides useful information.
code	How to indicate code.
kbd	How to show keyboard input.
key	How to specify keys.
samp	How to show a literal sequence of characters.
var	How to indicate a metasyntactic variable.
file	How to indicate the name of a file.
dfn	How to specify a definition.
cite	How to refer to a book that is not in Info.

1.100 texi.guide/Useful Highlighting

Highlighting Commands are Useful

The highlighting commands can be used to generate useful information from the file, such as lists of functions or file names. It is possible, for example, to write a program in Emacs Lisp (or a keyboard macro) to insert an index entry after every paragraph that contains words or phrases marked by a specified command. You could do this to

construct an index of functions if you had not already made the entries.

The commands serve a variety of purposes:

`'@code{SAMPLE-CODE}'`

Indicate text that is a literal example of a piece of a program.

`'@kbd{KEYBOARD-CHARACTERS}'`

Indicate keyboard input.

`'@key{KEY-NAME}'`

Indicate the conventional name for a key on a keyboard.

`'@samp{TEXT}'`

Indicate text that is a literal example of a sequence of characters.

`'@var{METASYNTACTIC-VARIABLE}'`

Indicate a metasyntactic variable.

`'@file{FILE-NAME}'`

Indicate the name of a file.

`'@dfn{TERM}'`

Indicate the introductory or defining use of a term.

`'@cite{REFERENCE}'`

Indicate the name of a book.

1.101 texi.guide/code

`'@code' {SAMPLE-CODE}`

Use the `'@code'` command to indicate text that is a piece of a program and which consists of entire syntactic tokens. Enclose the text in braces.

Thus, you should use `'@code'` for an expression in a program, for the name of a variable or function used in a program, or for a keyword. Also, you should use `'@code'` for the name of a program, such as `'diff'`, that is a name used in the machine. (You should write the name of a program in the ordinary text font if you regard it as a new English word, such as `'Emacs'` or `'Bison'`.)

Use `'@code'` for environment variables such as `'TEXINPUTS'`, and other variables.

Use `'@code'` for command names in command languages that resemble programming languages, such as Texinfo or the shell. For example, `'@code'` and `'@samp'` are produced by writing `'@code{@@code}'` and `'@code{@@samp}'` in the Texinfo source, respectively.

Note, however, that you should not use `'@code'` for shell options

such as `'-c'` when such options stand alone. (Use `'@samp'`.) Also, an entire shell command often looks better if written using `'@samp'` rather than `'@code'`. In this case, the rule is to choose the more pleasing format.

It is incorrect to alter the case of a word inside an `'@code'` command when it appears at the beginning of a sentence. Most computer languages are case sensitive. In C, for example, `'Printf'` is different from the identifier `'printf'`, and most likely is a misspelling of it. Even in languages which are not case sensitive, it is confusing to a human reader to see identifiers spelled in different ways. Pick one spelling and always use that. If you do not want to start a sentence with a command written all in lower case, you should rearrange the sentence.

Do not use the `'@code'` command for a string of characters shorter than a syntactic token. If you are writing about `'TEXINPUT'`, which is just a part of the name for the `'TEXINPUTS'` environment variable, you should use `'@samp'`.

In particular, you should not use the `'@code'` command when writing about the characters used in a token; do not, for example, use `'@code'` when you are explaining what letters or printable symbols can be used in the names of functions. (Use `'@samp'`.) Also, you should not use `'@code'` to mark text that is considered input to programs unless the input is written in a language that is like a programming language. For example, you should not use `'@code'` for the keystroke commands of GNU Emacs (use `'@kbd'` instead) although you may use `'@code'` for the names of the Emacs Lisp functions that the keystroke commands invoke.

In the printed manual, `'@code'` causes TeX to typeset the argument in a typewriter face. In the Info file, it causes the Info formatting commands to use single quotation marks around the text.

For example,

```
Use @code{diff} to compare two files.
```

produces this in the printed manual:

```
Use 'diff' to compare two files.
```

1.102 texi.guide/kbd

```
'@kbd' {KEYBOARD-CHARACTERS}
```

Use the `'@kbd'` command for characters of input to be typed by users. For example, to refer to the characters `'M-a'`, write

```
@kbd{M-a}
```

and to refer to the characters `'M-x shell'`, write

```
@kbd{M-x shell}
```

The `'@kbd'` command has the same effect as `'@code'` in Info, but may produce a different font in a printed manual.

You can embed another `@-`command inside the braces of an `'@kbd'` command. Here, for example, is the way to describe a command that would be described more verbosely as "press an `'r'` and then press the RET key":

```
@kbd{r @key{RET}}
```

This produces: `'r RET'`

You also use the `'@kbd'` command if you are spelling out the letters you type; for example:

```
To give the @code{logout} command,
type the characters @kbd{l o g o u t @key{RET}}.
```

This produces:

```
To give the 'logout' command, type the characters 'l o g o u t
RET'.
```

(Also, this example shows that you can add spaces for clarity. If you really want to mention a space character as one of the characters of input, write `'@key{SPC}'` for it.)

1.103 texi.guide/key

```
'@key' {KEY-NAME}
```

```
-----
```

Use the `'@key'` command for the conventional name for a key on a keyboard, as in:

```
@key{RET}
```

You can use the `'@key'` command within the argument of an `'@kbd'` command when the sequence of characters to be typed includes one or more keys that are described by name.

For example, to produce `'C-x ESC'` you would type:

```
@kbd{C-x @key{ESC}}
```

Here is a list of the recommended names for keys; they are all in upper case:

SPC

Space

RET

	Return
LFD	Linefeed
TAB	Tab
BS	Backspace
ESC	Escape
DEL	Delete
SFT	Shift
CTL	Control
META	Meta

There are subtleties to handling words like 'meta' or 'ctl' that are names of shift keys. When mentioning a character in which the shift key is used, such as 'Meta-a', use the '@kbd' command alone; do not use the '@key' command; but when you are referring to the shift key in isolation, use the '@key' command. For example, write '@kbd{Meta-a}' to produce 'Meta-a' and '@key{META}' to produce META. This is because 'Meta-a' refers to keys that you press on a keyboard, but META refers to a key without implying that you press it. In short, use '@kbd' for what you do, and use '@key' for what you talk about: "Press '@kbd{M-a}' to move point to the beginning of the sentence. The '@key{META}' key is often in the lower left of the keyboard."

1.104 texi.guide/samp

```
'@samp' {TEXT}
```

```
-----
```

Use the '@samp' command to indicate text that is a literal example or 'sample' of a sequence of characters in a file, string, pattern, etc. Enclose the text in braces. The argument appears within single quotation marks in both the Info file and the printed manual; in addition, it is printed in a fixed-width font.

To match @samp{foo} at the end of the line,
use the regexp @samp{foo\$}.

produces

To match `'foo'` at the end of the line, use the regexp `'foo$'`.

Any time you are referring to single characters, you should use `'@samp'` unless `'@kbd'` is more appropriate. Use `'@samp'` for the names of command-line options. Also, you may use `'@samp'` for entire statements in C and for entire shell commands--in this case, `'@samp'` often looks better than `'@code'`. Basically, `'@samp'` is a catchall for whatever is not covered by `'@code'`, `'@kbd'`, or `'@key'`.

Only include punctuation marks within braces if they are part of the string you are specifying. Write punctuation marks outside the braces if those punctuation marks are part of the English text that surrounds the string. In the following sentence, for example, the commas and period are outside of the braces:

```
In English, the vowels are @samp{a}, @samp{e},
@samp{i}, @samp{o}, @samp{u}, and sometimes
@samp{y}.
```

This produces:

```
In English, the vowels are 'a', 'e', 'i', 'o', 'u', and sometimes
'y'.
```

1.105 texi.guide/var

```
'@var' {METASYNTACTIC-VARIABLE}
```

Use the `'@var'` command to indicate metasyntactic variables. A "metasyntactic variable" is something that stands for another piece of text. For example, you should use a metasyntactic variable in the documentation of a function to describe the arguments that are passed to that function.

Do not use `'@var'` for the names of particular variables in programming languages. These are specific names from a program, so `'@code'` is correct for them. For example, the Lisp variable `'texinfo-tex-command'` is not a metasyntactic variable; it is properly formatted using `'@code'`.

The effect of `'@var'` in the Info file is to change the case of the argument to all upper case; in the printed manual, to italicize it.

For example,

```
To delete file @var{filename},
type @code{rm @var{filename}}.
```

produces

```
To delete file FILENAME, type 'rm FILENAME'.
```

(Note that `'@var'` may appear inside `'@code'`, `'@samp'`, `'@file'`, etc.)

Write a metasyntactic variable all in lower case without spaces, and use hyphens to make it more readable. Thus, the Texinfo source for the illustration of how to begin a Texinfo manual looks like this:

```
\input texinfo
@@setfilename @var{info-file-name}
@@settitle @var{name-of-manual}
```

This produces:

```
\input texinfo
@setfilename INFO-FILE-NAME
@settitle NAME-OF-MANUAL
```

In some documentation styles, metasyntactic variables are shown with angle brackets, for example:

```
..., type rm <filename>
```

However, that is not the style that Texinfo uses. (You can, of course, modify the sources to TeX and the Info formatting commands to output the '<...>' format if you wish.)

1.106 texi.guide/file

```
`@file' {FILE-NAME}
```

Use the '@file' command to indicate text that is the name of a file, buffer, or directory, or is the name of a node in Info. You can also use the command for file name suffixes. Do not use '@file' for symbols in a programming language; use '@code'.

Currently, '@file' is equivalent to '@samp' in its effects. For example,

```
The @file{.el} files are in
the @file{/gnu/emacs/lisp} directory.
```

produces

```
The '.el' files are in the '/gnu/emacs/lisp' directory.
```

1.107 texi.guide/dfn

```
`@dfn' {TERM}
```

Use the '@dfn' command to identify the introductory or defining use

of a technical term. Use the command only in passages whose purpose is to introduce a term which will be used again or which the reader ought to know. Mere passing mention of a term for the first time does not deserve '@dfn'. The command generates italics in the printed manual, and double quotation marks in the Info file. For example:

Getting rid of a file is called @dfn{deleting} it.

produces

Getting rid of a file is called "deleting" it.

As a general rule, a sentence containing the defining occurrence of a term should be a definition of the term. The sentence does not need to say explicitly that it is a definition, but it should contain the information of a definition--it should make the meaning clear.

1.108 texi.guide/cite

'@cite'{REFERENCE}

Use the '@cite' command for the name of a book that lacks a companion Info file. The command produces italics in the printed manual, and quotation marks in the Info file.

(If a book is written in Texinfo, it is better to use a cross reference command since a reader can easily follow such a reference in Info. See '@xref'.)

1.109 texi.guide/Emphasis

Emphasizing Text

=====

Usually, Texinfo changes the font to mark words in the text according to what category the words belong to; an example is the '@code' command. Most often, this is the best way to mark words. However, sometimes you will want to emphasize text without indicating a category. Texinfo has two commands to do this. Also, Texinfo has several commands that specify the font in which TeX will typeset text. These commands have no affect on Info and only one of them, the '@r' command, has any regular use.

emph & strong
Smallcaps
Fonts

How to emphasize text in Texinfo.
How to use the small caps font.
Various font commands for printed output.

1.110 texi.guide/emph & strong

`'@emph' {TEXT}` and `'@strong' {TEXT}`

The `'@emph'` and `'@strong'` commands are for emphasis; `'@strong'` is stronger. In printed output, `'@emph'` produces *italics* and `'@strong'` produces **bold**.

For example,

```
@quotation
@strong{Caution:} @code{rm * .[^.]*} removes @emph{all}
files in the directory.
@end quotation
```

produces:

```
*Caution*: `rm * .[^.]*' removes all
files in the directory.
```

The `'@strong'` command is seldom used except to mark what is, in effect, a typographical element, such as the word `'Caution'` in the preceding example.

In the Info file, both `'@emph'` and `'@strong'` put asterisks around the text.

```
*Caution:* Do not use '@emph' or '@strong' with the word 'Note';
Info will mistake the combination for a cross reference. Use a
phrase such as *Please note* or *Caution* instead.
```

1.111 texi.guide/Smallcaps

`'@sc' {TEXT}`: The Small Caps Font

Use the `'@sc'` command to set text in the printed output in a small caps font and set text in the Info file in upper case letters.

Write the text between braces in lower case, like this:

```
The @sc{acm} and @sc{ieee} are technical societies.
```

This produces:

```
The ACM and IEEE are technical societies.
```

TeX typesets the small caps font in a manner that prevents the letters from 'jumping out at you on the page'. This makes small caps text easier to read than text in all upper case. The Info formatting commands set all small caps text in upper case.

If the text between the braces of an '@sc' command is upper case, TeX typesets in full-size capitals. Use full-size capitals sparingly.

You may also use the small caps font for a jargon word such as ATO (a NASA word meaning 'abort to orbit').

There are subtleties to using the small caps font with a jargon word such as CDR, a word used in Lisp programming. In this case, you should use the small caps font when the word refers to the second and subsequent elements of a list (the CDR of the list), but you should use '@code' when the word refers to the Lisp function of the same spelling.

1.112 texi.guide/Fonts

Fonts for Printing, Not Info

Texinfo provides four font commands that specify font changes in the printed manual but have no effect in the Info file. '@i' requests italic font (in some versions of TeX, a slanted font is used), '@b' requests bold face, '@t' requests the fixed-width, typewriter-style font used by '@code', and '@r' requests a roman font, which is the usual font in which text is printed. All four commands apply to an argument that follows, surrounded by braces.

Only the '@r' command has much use: in example programs, you can use the '@r' command to convert code comments from the fixed-width font to a roman font. This looks better in printed output.

For example,

```
@lisp
(+ 2 2)      ; @r{Add two plus two.}
@end lisp
```

produces

```
(+ 2 2)      ; Add two plus two.
```

If possible, you should avoid using the other three font commands. If you need to use one, it probably indicates a gap in the Texinfo language.

1.113 texi.guide/Quotations and Examples

Quotations and Examples

Quotations and examples are blocks of text consisting of one or more whole paragraphs that are set off from the bulk of the text and treated

differently. They are usually indented.

In Texinfo, you always begin a quotation or example by writing an '@end' command that is also at the beginning of a line by itself. For instance, you begin an example by writing '@example' by itself at the beginning of a line and end the example by writing '@end example' on a line by itself, at the beginning of that line.

Block Enclosing Commands	Use different constructs for different purposes.
quotation	How to write a quotation.
example	How to write an example in a fixed-width font.
noindent	How to prevent paragraph indentation.
Lisp Example	How to illustrate Lisp code.
smallexample & smalllisp	Forms for the '@smallbook' option.
display	How to write an example in the current font.
format	How to write an example that does not narrow the margins.
exdent	How to undo the indentation of a line.
flushleft & flushright	How to push text flushleft or flushright.
cartouche	How to draw cartouches around examples.

1.114 texi.guide/Block Enclosing Commands

The Block Enclosing Commands

=====

Here are commands for quotations and examples:

'@quotation'

Indicate text that is quoted. The text is filled, indented, and printed in a roman font by default.

'@example'

Illustrate code, commands, and the like. The text is printed in a fixed-width font, and indented but not filled.

'@lisp'

Illustrate Lisp code. The text is printed in a fixed-width font, and indented but not filled.

'@smallexample'

Illustrate code, commands, and the like. Similar to '@example', except that in TeX this command typesets text in a smaller font for the smaller '@smallbook' format than for the 8.5 by 11 inch format.

'@smalllisp'

Illustrate Lisp code. Similar to '@lisp', except that in TeX this command typesets text in a smaller font for the smaller '@smallbook' format than for the 8.5 by 11 inch format.

'@display'

Display illustrative text. The text is indented but not filled, and no font is specified (so, by default, the font is roman).

`'@format'`

Print illustrative text. The text is not indented and not filled and no font is specified (so, by default, the font is roman).

The `'@exdent'` command is used within the above constructs to undo the indentation of a line.

The `'@flushleft'` and `'@flushright'` commands are used to line up the left or right margins of unfilled text.

The `'@noindent'` command may be used after one of the above constructs to prevent the following text from being indented as a new paragraph.

You can use the `'@cartouche'` command within one of the above constructs to highlight the example or quotation by drawing a box with rounded corners around it. (The `'@cartouche'` command affects only the printed manual; it has no effect in the Info file; see See Drawing Cartouches Around Examples.)

1.115 texi.guide/quotation

`'@quotation'`

=====

The text of a quotation is processed normally except that:

- * the margins are closer to the center of the page, so the whole of the quotation is indented;
- * the first lines of paragraphs are indented no more than other lines;
- * in the printed output, interparagraph spacing is reduced.

This is an example of text written between an `'@quotation'` command and an `'@end quotation'` command. An `'@quotation'` command is most often used to indicate text that is excerpted from another (real or hypothetical) printed work.

Write an `'@quotation'` command as text on a line by itself. This line will disappear from the output. Mark the end of the quotation with a line beginning with and containing only `'@end quotation'`. The `'@end quotation'` line will likewise disappear from the output. Thus, the following,

```
@quotation
This is
a foo.
@end quotation
```

produces

```
This is a foo.
```

1.116 texi.guide/example

```
'@example'  
=====
```

The '@example' command is used to indicate an example that is not part of the running text, such as computer input or output.

```
This is an example of text written between an  
'@example' command  
and an '@end example' command.  
The text is indented but not filled.
```

In the printed manual, the text is typeset in a fixed-width font, and extra spaces and blank lines are significant. In the Info file, an analogous result is obtained by indenting each line with five spaces.

Write an '@example' command at the beginning of a line by itself. This line will disappear from the output. Mark the end of the example with an '@end example' command, also written at the beginning of a line by itself. The '@end example' will disappear from the output.

For example,

```
@example  
mv foo bar  
@end example
```

produces

```
mv foo bar
```

Since the lines containing '@example' and '@end example' will disappear, you should put a blank line before the '@example' and another blank line after the '@end example'. (Remember that blank lines between the beginning '@example' and the ending '@end example' will appear in the output.)

***Caution:** Do not use tabs in the lines of an example (or anywhere else in Texinfo, for that matter)! TeX treats tabs as single spaces, and that is not what they look like. This is a problem with TeX. (If necessary, in Emacs, you can use 'M-x untabify' to convert tabs in a region to multiple spaces.)

Examples are often, logically speaking, "in the middle" of a paragraph, and the text continues after an example should not be indented. The '@noindent' command prevents a piece of text from being indented as if it were a new paragraph. (See noindent.)

(The `'@code'` command is used for examples of code that are embedded within sentences, not set off from preceding and following text. See `'@code'.`)

1.117 texi.guide/noindent

```
'@noindent'
=====
```

An example or other inclusion can break a paragraph into segments. Ordinarily, the formatters indent text that follows an example as a new paragraph. However, you can prevent this by writing `'@noindent'` at the beginning of a line by itself preceding the continuation text.

For example:

```
@example
This is an example
@end example
```

```
@noindent
This line is not indented. As you can see, the
beginning of the line is fully flush left with the line
that follows after it. (This whole example is between
@code{@@display} and @code{@@end display}.)
```

produces

```
This is an example
```

```
This line is not indented. As you can see, the
beginning of the line is fully flush left with the line
that follows after it. (This whole example is between
'@display' and '@end display'.)
```

To adjust the number of blank lines properly in the Info file output, remember that the line containing `'@noindent'` does not generate a blank line, and neither does the `'@end example'` line.

In the Texinfo source file for this manual, each line that says `'produces'` is preceded by a line containing `'@noindent'`.

Do not put braces after an `'@noindent'` command; they are not necessary, since `'@noindent'` is a command used outside of paragraphs (see Command Syntax).

1.118 texi.guide/Lisp Example

```
'@lisp'
=====
```

The '@lisp' command is used for Lisp code. It is synonymous with the '@example' command.

This is an example of text written between an '@lisp' command and an '@end lisp' command.

Use '@lisp' instead of '@example' so as to preserve information regarding the nature of the example. This is useful, for example, if you write a function that evaluates only and all the Lisp code in a Texinfo file. Then you can use the Texinfo file as a Lisp library.(1)

Mark the end of '@lisp' with '@end lisp' on a line by itself.

----- Footnotes -----

(1) It would be straightforward to extend Texinfo to work in a similar fashion for C, FORTRAN, or other languages.

1.119 texi.guide/smalleexample & smalllisp

```
'@smalleexample' and '@smalllisp'
=====
```

In addition to the regular '@example' and '@lisp' commands, Texinfo has two other "example-style" commands. These are the '@smalleexample' and '@smalllisp' commands. Both these commands are designed for use with the '@smallbook' command that causes TeX to produce a printed manual in a 7 by 9.25 inch format rather than the regular 8.5 by 11 inch format.

In TeX, the '@smalleexample' and '@smalllisp' commands typeset text in a smaller font for the smaller '@smallbook' format than for the 8.5 by 11 inch format. Consequently, many examples containing long lines fit in a narrower, '@smallbook' page without needing to be shortened. Both commands typeset in the normal font size when you format for the 8.5 by 11 inch size; indeed, in this situation, the '@smalleexample' and '@smalllisp' commands are defined to be the '@example' and '@lisp' commands.

In Info, the '@smalleexample' and '@smalllisp' commands are equivalent to the '@example' and '@lisp' commands, and work exactly the same.

Mark the end of '@smalleexample' or '@smalllisp' with '@end smalleexample' or '@end smalllisp', respectively.

This is an example of text written between '@smalleexample' and '@end smalleexample'. In Info and in an 8.5 by 11 inch manual, this text appears in its normal size; but in a 7 by 9.25 inch manual, this text appears in a smaller font.

The `'@smallexample'` and `'@smalllisp'` commands make it easier to prepare smaller format manuals without forcing you to edit examples by hand to fit them onto narrower pages.

As a general rule, a printed document looks better if you write all the examples in a chapter consistently in `'@example'` or in `'@smallexample'`. Only occasionally should you mix the two formats.

See `Printing` , for more information about the `'@smallbook'` command.

1.120 texi.guide/display

`'@display'`
=====

The `'@display'` command begins a kind of example. It is like the `'@example'` command except that, in a printed manual, `'@display'` does not select the fixed-width font. In fact, it does not specify the font at all, so that the text appears in the same font it would have appeared in without the `'@display'` command.

This is an example of text written between an `'@display'` command and an `'@end display'` command. The `'@display'` command indents the text, but does not fill it.

1.121 texi.guide/format

`'@format'`
=====

The `'@format'` command is similar to `'@example'` except that, in the printed manual, `'@format'` does not select the fixed-width font and does not narrow the margins.

This is an example of text written between an `'@format'` command and an `'@end format'` command. As you can see from this example, the `'@format'` command does not fill the text.

1.122 texi.guide/exdent

`'@exdent'`: Undoing a Line's Indentation
=====

The `'@exdent'` command removes any indentation a line might have.

The command is written at the beginning of a line and applies only to the text that follows the command that is on the same line. Do not use braces around the text. In a printed manual, the text on an '@exdent' line is printed in the roman font.

'@exdent' is usually used within examples. Thus,

```
@example
This line follows an @@example command.
@exdent This line is exdented.
This line follows the exdented line.
The @@end example comes on the next line.
@end group
```

produces

```
This line follows an @example command.
This line is exdented.
This line follows the exdented line.
The @end example comes on the next line.
```

In practice, the '@exdent' command is rarely used. Usually, you un-indent text by ending the example and returning the page to its normal width.

1.123 texi.guide/flushleft & flushright

'@flushleft' and '@flushright'
=====

The '@flushleft' and '@flushright' commands line up the ends of lines on the left and right margins of a page, but do not fill the text. The commands are written on lines of their own, without braces. The '@flushleft' and '@flushright' commands are ended by '@end flushleft' and '@end flushright' commands on lines of their own.

For example,

```
@flushleft
This text is
written flushleft.
@end flushleft
```

produces

```
This text is
written flushleft.
```

Flushright produces the type of indentation often used in the return address of letters.

For example,

```
@flushright
```

```

Here is an example of text written
flushright.  The @code{@flushright} command
right justifies every line but leaves the
left end ragged.
@end flushright

```

produces

```

                Here is an example of text written
                flushright.  The '@flushright' command
right justifies every line but leaves the
                                left end ragged.

```

1.124 texi.guide/cartouche

Drawing Cartouches Around Examples

=====

In a printed manual, the '@cartouche' command draws a box with rounded corners around its contents. You can use this command to further highlight an example or quotation. For instance, you could write a manual in which one type of example is surrounded by a cartouche for emphasis.

The '@cartouche' command affects only the printed manual; it has no effect in the Info file.

For example,

```

@example
@cartouche
% pwd
/gnu/lib/emacs/info
@end cartouche
@end example

```

surrounds the two-line example with a box with rounded corners, in the printed manual.

1.125 texi.guide/Lists and Tables

Making Lists and Tables

Texinfo has several ways of making lists and two-column tables. Lists can be bulleted or numbered, while two-column tables can highlight the items in the first column.

Introducing Lists

Texinfo formats lists for you.

itemize	How to construct a simple list.
enumerate	How to construct a numbered list.
Two-column Tables	How to construct a two-column table.

1.126 texi.guide/Introducing Lists

Introducing Lists
=====

Texinfo automatically indents the text in lists or tables, and numbers an enumerated list. This last feature is useful if you modify the list, since you do not need to renumber it yourself.

Numbered lists and tables begin with the appropriate @-command at the beginning of a line, and end with the corresponding '@end' command on a line by itself. The table and itemized-list commands also require that you write formatting information on the same line as the beginning

Begin an enumerated list, for example, with an '@enumerate' command and end the list with an '@end enumerate' command. Begin an itemized list with an '@itemize' command, followed on the same line by a formatting command such as '@bullet', and end the list with an '@end itemize' command.

Precede each element of a list with an '@item' or '@itemx' command.

Here is an itemized list of the different kinds of table and lists:

- * Itemized lists with and without bullets.
- * Enumerated lists, using numbers or letters.
- * Two-column tables with highlighting.

Here is an enumerated list with the same items:

1. Itemized lists with and without bullets.
2. Enumerated lists, using numbers or letters.
3. Two-column tables with highlighting.

And here is a two-column table with the same items and their @-commands:

@itemize'	Itemized lists with and without bullets.
@enumerate'	Enumerated lists, using numbers or letters.
@table'	
@ftable'	
@vtable'	Two-column tables with highlighting.

1.127 texi.guide/itemize

Making an Itemized List

=====

The '@itemize' command produces sequences of indented paragraphs, with a bullet or other mark inside the left margin at the beginning of each paragraph for which such a mark is desired.

Begin an itemized list by writing '@itemize' at the beginning of a line. Follow the command, on the same line, with a character or a Texinfo command that generates a mark. Usually, you will write '@bullet' after '@itemize', but you can use '@minus', or any character or any special symbol that results in a single character in the Info file. (When you write '@bullet' or '@minus' after an '@itemize' command, you may omit the '{}'.)

Write the text of the indented paragraphs themselves after the '@itemize', up to another line that says '@end itemize'.

Before each paragraph for which a mark in the margin is desired, write a line that says just '@item'. Do not write any other text on this line.

Usually, you should put a blank line before an '@item'. This puts a blank line in the Info file. (TeX inserts the proper interline whitespace in either case.) Except when the entries are very brief, these blank lines make the list look better.

Here is an example of the use of '@itemize', followed by the output it produces. Note that '@bullet' produces an '*' in Info and a round dot in TeX.

```
@itemize @bullet
@item
Some text for foo.

@item
Some text
for bar.
@end itemize
```

This produces:

```
* Some text for foo.

* Some text for bar.
```

Itemized lists may be embedded within other itemized lists. Here is a list marked with dashes embedded in a list marked with bullets:

```
@itemize @bullet
@item
```

```

First item.

@itemize @minus
@item
Inner item.

@item
Second inner item.
@end itemize

@item
Second outer item.
@end itemize

```

This produces:

```

* First item.

  - Inner item.

  - Second inner item.

* Second outer item.

```

1.128 texi.guide/enumerate

Making a Numbered or Lettered List
=====

'@enumerate' is like '@itemize' except that the marks in the left margin contain successive integers or letters. (See '@itemize'.)

Write the '@enumerate' command at the beginning of a line. The command does not require an argument, but accepts either a number or a letter as an option. Without an argument, '@enumerate' starts the list with the number 1. With a numeric argument, such as 3, the command starts the list with that number. With an upper or lower case letter, such as 'a' or 'A', the command starts the list with that letter.

Write the text of the enumerated list in the same way you write an itemized list: put '@item' on a line of its own before the start of each paragraph that you want enumerated. Do not write any other text on the line beginning with '@item'.

You should put a blank line between entries in the list. This generally makes it easier to read the Info file.

Here is an example of '@enumerate' without an argument:

```

@enumerate
@item
Underlying causes.

@item

```

```
Proximate causes.  
@end enumerate
```

This produces:

1. Underlying causes.
2. Proximate causes.

Here is an example with an argument of '3':

```
@enumerate 3  
@item  
Predisposing causes.  
  
@item  
Precipitating causes.  
  
@item  
Perpetuating causes.  
@end enumerate
```

This produces:

3. Predisposing causes.
4. Precipitating causes.
5. Perpetuating causes.

Here is a brief summary of the alternatives. The summary is constructed using '@enumerate' with an argument of 'a'.

a. '@enumerate'

Without an argument, produce a numbered list, starting with the number 1.

b. '@enumerate POSITIVE-INTEGER'

With a (positive) numeric argument, start a numbered list with that number. You can use this to continue a list that you interrupted with other text.

c. '@enumerate UPPER-CASE-LETTER'

With an upper case letter as argument, start a list in which each item is marked by a letter, beginning with that upper case letter.

d. '@enumerate LOWER-CASE-LETTER'

With a lower case letter as argument, start a list in which each item is marked by a letter, beginning with that lower case letter.

You can also nest enumerated lists, as in an outline.

1.129 texi.guide/Two-column Tables

Making a Two-column Table

=====

'@table' is similar to '@itemize', but the command allows you to specify a name or heading line for each item. (See '@itemize'.) The '@table' command is used to produce two-column tables, and is especially useful for glossaries and explanatory exhibits.

table	How to construct a two-column table.
ftable vtable	How to construct a two-column table with automatic indexing.
itemx	How to put more entries in the first column.

1.130 texi.guide/table

Using the '@table' Command

Use the '@table' command to produce two-column tables.

Write the '@table' command at the beginning of a line and follow it on the same line with an argument that is a Texinfo command such as '@code', '@samp', '@var', or '@kbd'. Although these commands are usually followed by arguments in braces, in this case you use the command name without an argument because '@item' will supply the argument. This command will be applied to the text that goes into the first column of each item and determines how it will be highlighted. For example, '@samp' will cause the text in the first column to be highlighted with an '@samp' command.

You may also choose to use the '@asis' command as an argument to '@table'. '@asis' is a command that does nothing; if you use this command after '@table', TeX and the Info formatting commands output the first column entries without added highlighting ('as is').

(The '@table' command may work with other commands besides those listed here. However, you can only use commands that normally take arguments in braces.)

Begin each table entry with an '@item' command at the beginning of a line. Write the first column text on the same line as the '@item' command. Write the second column text on the line following the '@item' line and on subsequent lines. (You do not need to type anything for an empty second column entry.) You may write as many lines of supporting text as you wish, even several paragraphs. But only text on the same line as the '@item' will be placed in the first column.

Normally, you should put a blank line before an '@item' line. This puts a blank line in the Info file. Except when the entries are very

brief, a blank line looks better.

The following table, for example, highlights the text in the first column with an '@samp' command:

```
@table @samp
@item foo
This is the text for
@samp{foo}.

@item bar
Text for @samp{bar}.
@end table
```

This produces:

```
'foo'
  This is the text for 'foo'.

'bar'
  Text for 'bar'.
```

If you want to list two or more named items with a single block of text, use the '@itemx' command. (See '@itemx'.)

1.131 texi.guide/ftable vtable

'@ftable' and '@vtable'

The '@ftable' and '@vtable' commands are the same as the '@table' command except that '@ftable' automatically enters each of the items in the first column of the table into the index of functions and '@vtable' automatically enters each of the items in the first column of the table into the index of variables. This simplifies the task of creating indices. Only the items on the same line as the '@item' commands are indexed, and they are indexed in exactly the form that they appear on that line. See Creating Indices, for more information about indices.

Begin a two-column table using '@ftable' or '@vtable' by writing the argument that is a Texinfo command such as '@code', exactly as you would for an '@table' command; and end the table with an '@end ftable' or '@end vtable' command on a line by itself.

1.132 texi.guide/itemx

'@itemx'

Use the '@itemx' command inside a table when you have two or more

first column entries for the same item, each of which should appear on a line of its own. Use '@itemx' for all but the first entry. The '@itemx' command works exactly like '@item' except that it does not generate extra vertical space above the first column text.

For example,

```
@table @code
@item upcase
@itemx downcase
These two functions accept a character or a string as
argument, and return the corresponding upper case (lower
case) character or string.
@end table
```

This produces:

```
'upcase'
'downcase'
  These two functions accept a character or a string as argument,
  and return the corresponding upper case (lower case) character or
  string.
```

(Note also that this example illustrates multi-line supporting text in a two-column table.)

1.133 texi.guide/Indices

Creating Indices

Using Texinfo, you can generate indices without having to sort and collate entries manually. In an index, the entries are listed in alphabetical order, together with information on how to find the discussion of each entry. In a printed manual, this information consists of page numbers. In an Info file, this information is a menu entry leading to the first node referenced.

Texinfo provides several predefined kinds of index: an index for functions, an index for variables, an index for concepts, and so on. You can combine indices or use them for other than their canonical purpose. If you wish, you can define your own indices.

Index Entries	Choose different words for index entries.
Predefined Indices	Use different indices for different kinds of entry.
Indexing Commands	How to make an index entry.
Combining Indices	How to combine indices.
New Indices	How to define your own indices.

1.134 texi.guide/Index Entries

Making Index Entries

=====

When you are making index entries, it is good practice to think of the different ways people may look for something. Different people *do not* think of the same words when they look something up. A helpful index will have items indexed under all the different words that people may use. For example, one reader may think it obvious that the two-letter names for indices should be listed under "Indices, two-letter names", since the word "Index" is the general concept. But another reader may remember the specific concept of two-letter names and search for the entry listed as "Two letter names for indices". A good index will have both entries and will help both readers.

Like typesetting, the construction of an index is a highly skilled, professional art, the subtleties of which are not appreciated until you need to do it yourself.

See *Printing Indices & Menus*, for information about printing an index at the end of a book or creating an index menu in an Info file.

1.135 texi.guide/Predefined Indices

Predefined Indices

=====

Texinfo provides six predefined indices:

- * A "concept index" listing concepts that are discussed.
- * A "function index" listing functions (such as entry points of libraries).
- * A "variables index" listing variables (such as global variables of libraries).
- * A "keystroke index" listing keyboard commands.
- * A "program index" listing names of programs.
- * A "data type index" listing data types (such as structures defined in header files).

Not every manual needs all of these, and most manuals use two or three of them. This manual has two indices: a concept index and an @-command index (that is actually the function index but is called a command index in the chapter heading). Two or more indices can be combined into one using the '@synindex' or '@syncodeindex' commands. See *Combining Indices*.

1.136 texi.guide/Indexing Commands

Defining the Entries of an Index

=====

The data to make an index come from many individual indexing commands scattered throughout the Texinfo source file. Each command says to add one entry to a particular index; after formatting, the index will give the current page number or node name as the reference.

An index entry consists of an indexing command at the beginning of a line followed, on the rest of the line, by the entry.

For example, this section begins with the following five entries for the concept index:

```
@cindex Defining indexing entries
@cindex Index entries
@cindex Entries for an index
@cindex Specifying index entries
@cindex Creating index entries
```

Each predefined index has its own indexing command--`@cindex' for the concept index, `@findex' for the function index, and so on.

The usual convention is to capitalize the first word of each index entry, unless that word is the name of a function, variable, or other such entity that should not be capitalized. Thus, if you are documenting Emacs Lisp, you should usually capitalize entries in the concept index, but not those in the function index. However, if your concept index entries are consistently short (one or two words each) it may look better for each regular entry to start with a lower case letter. Whichever convention you adapt, please be consistent!

By default, entries for a concept index are printed in a small roman font and entries for the other indices are printed in a small `@code' font. You may change the way part of an entry is printed with the usual Texinfo commands, such as `@file' for file names and `@emph' for emphasis (see Marking Text).

The six indexing commands for predefined indices are:

```
`@cindex CONCEPT'
  Make an entry in the concept index for CONCEPT.

`@findex FUNCTION'
  Make an entry in the function index for FUNCTION.

`@vindex VARIABLE'
  Make an entry in the variable index for VARIABLE.

`@kindex KEYSTROKE'
  Make an entry in the key index for KEYSTROKE.

`@pindex PROGRAM'
  Make an entry in the program index for PROGRAM.
```

`'@tindex DATA TYPE'`

Make an entry in the data type index for DATA TYPE.

**Caution:* Do not use a colon in an index entry. In Info, a colon separates the menu entry name from the node name. An extra colon confuses Info. See The Parts of a Menu, for more information about the structure of a menu entry.

If you write several identical index entries in different places in a Texinfo file, the index in the printed manual will list all the pages to which those entries refer. However, the index in the Info file will list *only* the node that references the *first* of those index entries. Therefore, it is best to write indices in which each entry refers to only one place in the Texinfo file. Fortunately, this constraint is a feature rather than a loss since it means that the index will be easy to use. Otherwise, you could create an index that lists several pages for one entry and your reader would not know to which page to turn. If you have two identical entries for one topic, change the topics slightly, or qualify them to indicate the difference.

You are not actually required to use the predefined indices for their canonical purposes. For example, suppose you wish to index some C preprocessor macros. You could put them in the function index along with actual functions, just by writing `'@findex'` commands for them; then, when you print the "Function Index" as an unnumbered chapter, you could give it the title 'Function and Macro Index' and all will be consistent for the reader. Or you could put the macros in with the data types by writing `'@tindex'` commands for them, and give that index a suitable title so the reader will understand. (See Printing Indices & Menus.)

1.137 texi.guide/Combining Indices

Combining Indices

=====

Sometimes you will want to combine two disparate indices such as functions and concepts, perhaps because you have few enough of one of them that a separate index for them would look silly.

You could put functions into the concept index by writing `'@cindex'` commands for them instead of `'@findex'` commands, and produce a consistent manual by printing the concept index with the title 'Function and Concept Index' and not printing the 'Function Index' at all; but this is not a robust procedure. It works only if your document is never included as part of another document that is designed to have a separate function index; if your document were to be included with such a document, the functions from your document and those from the other would not end up together. Also, to make your function names appear in the right font in the concept index, you would need to enclose every one of them between the braces of `'@code'`.

<code>syncodeindex</code>	How to merge two indices, using <code>'@code'</code> font for the merged-from index.
<code>synindex</code>	How to merge two indices, using the default font of the merged-to index.

1.138 texi.guide/syncodeindex

`'@syncodeindex'`

When you want to combine functions and concepts into one index, you should index the functions with `'@findex'` and index the concepts with `'@cindex'`, and use the `'@syncodeindex'` command to redirect the function index entries into the concept index.

The `'@syncodeindex'` command takes two arguments; they are the name of the index to redirect, and the name of the index to redirect it to. The template looks like this:

```
@syncodeindex FROM TO
```

For this purpose, the indices are given two-letter names:

```
'cp'
  concept index

'fn'
  function index

'vr'
  variable index

'ky'
  key index

'pg'
  program index

'tp'
  data type index
```

Write an `'@syncodeindex'` command before or shortly after the end-of-header line at the beginning of a Texinfo file. For example, to merge a function index with a concept index, write the following:

```
@syncodeindex fn cp
```

This will cause all entries designated for the function index to merge in with the concept index instead.

To merge both a variables index and a function index into a concept index, write the following:

```
@syncodeindex vr cp
```

```
@syncodeindex fn cp
```

The `'@syncodeindex'` command puts all the entries from the `'from'` index (the redirected index) into the `'@code'` font, overriding whatever default font is used by the index to which the entries are now directed. This way, if you direct function names from a function index into a concept index, all the function names are printed in the `'@code'` font as you would expect.

1.139 texi.guide/synindex

```
'@synindex'
.....
```

The `'@synindex'` command is nearly the same as the `'@syncodeindex'` command, except that it does not put the `'from'` index entries into the `'@code'` font; rather it puts them in the roman font. Thus, you use `'@synindex'` when you merge a concept index into a function index.

See *Printing Indices & Menus*, for information about printing an index at the end of a book or creating an index menu in an Info file.

1.140 texi.guide/New Indices

```
Defining New Indices
=====
```

In addition to the predefined indices, you may use the `'@defindex'` and `'@defcodeindex'` commands to define new indices. These commands create new indexing `@`-commands with which you mark index entries. The `'@defindex'` command is used like this:

```
@defindex NAME
```

The name of an index should be a two letter word, such as `'au'`. For example:

```
@defindex au
```

This defines a new index, called the `'au'` index. At the same time, it creates a new indexing command, `'@auindex'`, that you can use to make index entries. Use the new indexing command just as you would use a predefined indexing command.

For example, here is a section heading followed by a concept index entry and two `'au'` index entries.

```
@section Cognitive Semantics
@cindex kinesthetic image schemas
@auindex Johnson, Mark
```

```
@auindex Lakoff, George
```

(Evidently, 'au' serves here as an abbreviation for "author".) Texinfo constructs the new indexing command by concatenating the name of the index with 'index'; thus, defining an 'au' index leads to the automatic creation of an '@auindex' command.

Use the '@printindex' command to print the index, as you do with the predefined indices. For example:

```
@node Author Index, Subject Index, , Top
@unnumbered Author Index
```

```
@printindex au
```

The '@defcodeindex' is like the '@defindex' command, except that, in the printed output, it prints entries in an '@code' font instead of a roman font. Thus, it parallels the '@findex' command rather than the '@cindex' command.

You should define new indices within or right after the end-of-header line of a Texinfo file, before any '@synindex' or '@syncodeindex' commands (see Header).

1.141 texi.guide/Insertions

Special Insertions

```
*****
```

Texinfo provides several commands for formatting dimensions, for inserting single characters that have special meaning in Texinfo, such as braces, and for inserting special graphic symbols that do not correspond to characters, such as dots and bullets.

Braces Atsigns Periods	How to insert braces, '@' and periods.
dmn	How to format a dimension.
Dots Bullets	How to insert dots and bullets.
TeX and copyright	How to insert the TeX logo and the copyright symbol.
minus	How to insert a minus sign.

1.142 texi.guide/Braces Atsigns Periods

Inserting '@', Braces, and Periods

```
=====
```

'@' and curly braces are special characters in Texinfo. To insert these characters so they appear in text, you must put an '@' in front of these characters to prevent Texinfo from misinterpreting them.

Periods are also special. Depending on whether the period is inside or at the end of a sentence, less or more space is inserted after a period in a typeset manual. Since it is not always possible for Texinfo to determine when a period ends a sentence and when it is used in an abbreviation, special commands are needed in some circumstances. (Usually, Texinfo can guess how to handle periods, so you do not need to use the special commands; you just enter a period as you would if you were using a typewriter, which means you put two spaces after the period, question mark, or exclamation mark that ends a sentence.)

Do not put braces after any of these commands; they are not necessary.

Inserting An Atsign

Inserting Braces

Controlling Spacing

How to insert '{' and '}'

How to insert the right amount of space
after punctuation within a sentence.

1.143 texi.guide/Inserting An Atsign

Inserting '@' with @@

'@@' stands for a single '@' in either printed or Info output.

Do not put braces after an '@@' command.

1.144 texi.guide/Inserting Braces

Inserting '{' and '}' with @{ and @}

,

'@}' stands for a single '}' in either printed or Info output.

Do not put braces after either an '{' command.

1.145 texi.guide/Controlling Spacing

Spacing After Colons and Periods

Use the '@:' command after a period, question mark, exclamation mark, or colon that should not be followed by extra space. For

example, use '@:' after periods that end abbreviations which are not at the ends of sentences. '@:' has no effect on the Info file output.

For example,

```
The s.o.p.@: has three parts ...
The s.o.p. has three parts ...
```

produces

```
The s.o.p. has three parts ...
The s.o.p. has three parts ...
```

@:' has no effect on the Info output. ('s.o.p' is an acronym for "Standard Operating Procedure".)

Use '@.' instead of a period at the end of a sentence that ends with a single capital letter. Otherwise, TeX will think the letter is an abbreviation and will not insert the correct end-of-sentence spacing. Here is an example:

```
Give it to M.I.B. and to M.E.W@. Also, give it to R.J.C@.
Give it to M.I.B. and to M.E.W. Also, give it to R.J.C.
```

produces

```
Give it to M.I.B. and to M.E.W Also, give it to R.J.C
Give it to M.I.B. and to M.E.W. Also, give it to R.J.C.
```

In the Info file output, '@.' is equivalent to a simple '.'.

The meanings of '@:' and '@.' in Texinfo are designed to work well with the Emacs sentence motion commands. This made it necessary for them to be incompatible with some other formatting systems that use Do not put braces after either an '@:' or an '@.' command.

1.146 texi.guide/dmn

@dmn{DIMENSION}: Format a Dimension
=====

At times, you may want to write '12pt' or '8.5in' with little or no space between the number and the abbreviation for the dimension. You can use the '@dmn' command to do this. On seeing the command, TeX inserts just enough space for proper typesetting; the Info formatting commands insert no space at all, since the Info file does not require it.

To use the '@dmn' command, write the number and then follow it immediately, with no intervening space, by '@dmn', and then by the dimension within braces.

For example,

A4 paper is 8.27@dmn{in} wide.

produces

A4 paper is 8.27in wide.

Not everyone uses this style. Instead of writing `'8.27@dmn{in}'` in the Texinfo file, you may write `'8.27 in.'` or `'8.27 inches'`. (In these cases, the formatters may insert a line break between the number and the dimension. Also, if you write a period after an abbreviation within a sentence, you should write `'@:.'` after the period to prevent TeX from inserting extra whitespace. See Spacing After Colons and Periods.)

1.147 texi.guide/Dots Bullets

Inserting Ellipsis, Dots, and Bullets

=====

An "ellipsis" (a line of dots) is not typeset as a string of periods, so a special command is used for ellipsis in Texinfo. The `'@bullet'` command is special, too. Each of these commands is followed by a pair of braces, `'{}'`, without any whitespace between the name of the command and the braces. (You need to use braces with these commands because you can use them next to other text; without the braces, the formatters would be confused. See @-Command Syntax, for further information.)

dots	How to insert dots ...
bullet	How to insert a bullet.

1.148 texi.guide/dots

`'@dots' {}`

Use the `'@dots{}'` command to generate an ellipsis, which is three dots in a row, appropriately spaced, like this: `'...'`. Do not simply write three periods in the input file; that would work for the Info file output, but would produce the wrong amount of space between the periods in the printed manual.

1.149 texi.guide/bullet

`'@bullet' {}`

Use the `'@bullet{}`' command to generate a large round dot, or the closest possible thing to one. In Info, an asterisk is used.

Here is a bullet: *

When you use `'@bullet'` in `'@itemize'`, you do not need to type the braces, because `'@itemize'` supplies them. See `itemize`.

1.150 texi.guide/TeX and copyright

Inserting TeX and the Copyright Symbol

=====

The logo `'TeX'` is typeset in a special fashion and it needs an these commands is followed by a pair of braces, `'{}'`, without any whitespace between the name of the command and the braces.

<code>tex</code>	How to insert the TeX logo.
<code>copyright symbol</code>	How to use <code>'@copyright{}</code> '.

1.151 texi.guide/tex

`'@TeX{}`'

Use the `'@TeX{}`' command to generate `'TeX'`. In a printed manual, this is a special logo that is different from three ordinary letters. In Info, it just looks like `'TeX'`. The `'@TeX{}`' command is unique among Texinfo commands in that the T and the X are in upper case.

1.152 texi.guide/copyright symbol

`'@copyright{}`'

Use the `'@copyright{}`' command to generate `'(C)'`. In a printed manual, this is a `'c'` inside a circle, and in Info, this is `'(C)'`.

1.153 texi.guide/minus

``@minus{}``: Inserting a Minus Sign

=====

Use the ``@minus{}`` command to generate a minus sign. In a fixed-width font, this is a single hyphen, but in a proportional font, the symbol is the customary length for a minus sign--a little longer than a hyphen.

You can compare the two forms:

``-'` is a minus sign generated with ``@minus{}``,

``-'` is a hyphen generated with the character ``-'`.

In the fixed-width font used by Info, ``@minus{}`` is the same as a hyphen.

You should not use ``@minus{}`` inside ``@code'` or ``@example'` because the width distinction is not made in the fixed-width font they use.

When you use ``@minus'` to specify the mark beginning each entry in an itemized list, you do not need to type the braces (see `itemize`).

1.154 texi.guide/Glyphs

Glyphs for Examples

In Texinfo, code is often illustrated in examples that are delimited by ``@example'` and ``@end example'`, or by ``@lisp'` and ``@end lisp'`. In such examples, you can indicate the results of evaluation or an expansion using ``=>'` or ``==>'`. Likewise, there are commands to insert glyphs to indicate printed output, error messages, equivalence of expressions, and the location of point.

The glyph-insertion commands do not need to be used within an example, but most often they are. Every glyph-insertion command is followed by a pair of left- and right-hand braces.

Glyphs Summary

result	How to show the result of expression.
expansion	How to indicate an expansion.
Print Glyph	How to indicate printed output.
Error Glyph	How to indicate an error message.
Equivalence	How to indicate equivalence.
Point Glyph	How to indicate the location of point.

1.155 texi.guide/Glyphs Summary

Glyphs Summary

=====

Here are the different glyph commands:

=>

'@result{}' points to the result of an expression.

==>

'@expansion{}' shows the results of a macro expansion.

-|

'@print{}' indicates printed output.

error-->

'@error{}' indicates that the following text is an error message.

==

'@equiv{}' indicates the exact equivalence of two forms.

!-

'@point{}' shows the location of point.

1.156 texi.guide/result

=>: Indicating Evaluation

=====

Use the '@result{}' command to indicate the result of evaluating an expression.

The '@result{}' command is displayed as '=>' in Info and as a double stemmed arrow in the printed output.

Thus, the following,

```
(cdr '(1 2 3))
=> (2 3)
```

may be read as "'(cdr '(1 2 3))' evaluates to '(2 3)'".

1.157 texi.guide/expansion

==>: Indicating an Expansion

=====

When an expression is a macro call, it expands into a new expression. You can indicate the result of the expansion with the '@expansion{}' command.

The `'@expansion{'` command is displayed as `'==>'` in Info and as a long arrow with a flat base in the printed output.

For example, the following

```
@lisp
(third '(a b c))
  @expansion{} (car (cdr (cdr '(a b c))))
  @result{} c
@end lisp
```

produces

```
(third '(a b c))
  ==> (car (cdr (cdr '(a b c))))
  => c
```

which may be read as:

`'(third '(a b c))'` expands to `'(car (cdr (cdr '(a b c))))'`; the result of evaluating the expression is `'c'`.

Often, as in this case, an example looks better if the `'@expansion{'` and `'@result{'` commands are indented five spaces.

1.158 texi.guide/Print Glyph

`-|`: Indicating Printed Output
=====

Sometimes an expression will print output during its execution. You can indicate the printed output with the `'@print{'` command.

The `'@print{'` command is displayed as `'-|'` in Info and similarly, as a horizontal dash butting against a vertical bar, in the printed output.

In the following example, the printed text is indicated with `'-|'`, and the value of the expression follows on the last line.

```
(progn (print 'foo) (print 'bar))
  -| foo
  -| bar
  => bar
```

In a Texinfo source file, this example is written as follows:

```
@lisp
(progn (print 'foo) (print 'bar))
  @print{} foo
  @print{} bar
  @result{} bar
@end lisp
```

1.159 texti.guide/Error Glyph

`error-->`: Indicating an Error Message
=====

A piece of code may cause an error when you evaluate it. You can designate the error message with the `'@error{}`' command.

The `'@error{}`' command is displayed as `'error-->'` in Info and as the word `'error'` in a box in the printed output.

Thus,

```
@lisp
(+ 23 'x)
@error{} Wrong type argument: integer-or-marker-p, x
@end lisp
```

produces

```
(+ 23 'x)
error--> Wrong type argument: integer-or-marker-p, x
```

This indicates that the following error message is printed when you evaluate the expression:

```
Wrong type argument: integer-or-marker-p, x
```

Note that `'error-->'` itself is not part of the error message.

1.160 texti.guide/Equivalence

`==`: Indicating Equivalence
=====

Sometimes two expressions produce identical results. You can indicate the exact equivalence of two forms with the `'@equiv{}`' command.

The `'@equiv{}`' command is displayed as `'=='` in Info and as a three parallel horizontal lines in the printed output.

Thus,

```
@lisp
(make-sparse-keymap) @equiv{} (list 'keymap)
@end lisp
```

produces

```
(make-sparse-keymap) == (list 'keymap)
```

This indicates that evaluating `'(make-sparse-keymap)'` produces identical results to evaluating `'(list 'keymap)'`.

1.161 texti.guide/Point Glyph

Indicating Point in a Buffer
=====

Sometimes you need to show an example of text in an Emacs buffer. In such examples, the convention is to include the entire contents of the buffer in question between two lines of dashes containing the buffer name.

You can use the `'@point{}` command to show the location of point in the text in the buffer. (The symbol for point, of course, is not part of the text in the buffer; it indicates the place *between* two characters where point is located.)

The `'@point{}` command is displayed as `'-!-'` in Info and as a small five pointed star in the printed output.

The following example shows the contents of buffer `'foo'` before and after evaluating a Lisp command to insert the word `'changed'`.

```
----- Buffer: foo -----
This is the -!-contents of foo.
----- Buffer: foo -----

(insert "changed ")
=> nil
----- Buffer: foo -----
This is the changed -!-contents of foo.
----- Buffer: foo -----
```

In a Texinfo source file, the example is written like this:

```
@example
----- Buffer: foo -----
This is the @point{}contents of foo.
----- Buffer: foo -----

(insert "changed ")
@result{} nil
----- Buffer: foo -----
This is the changed @point{}contents of foo.
----- Buffer: foo -----
@end example
```

1.162 texi.guide/Breaks

Making and Preventing Breaks

Usually, a Texinfo file is processed both by TeX and by one of the Info formatting commands. Line, paragraph, or page breaks sometimes occur in the ‘wrong’ place in one or other form of output. You must ensure that text looks right both in the printed manual and in the Info file.

For example, in a printed manual, page breaks may occur awkwardly in the middle of an example; to prevent this, you can hold text together using a grouping command that keeps the text from being split across two pages. Conversely, you may want to force a page break where none would occur normally. Fortunately, problems like these do not often arise. When they do, use the break, break prevention, or pagination commands.

Break Commands	Cause and prevent splits.
Line Breaks	How to force a single line to use two lines.
w	How to prevent unwanted line breaks.
sp	How to insert blank lines.
page	How to force the start of a new page.
group	How to prevent unwanted page breaks.
need	Another way to prevent unwanted page breaks.

1.163 texi.guide/Break Commands

The Break Commands

=====

The break commands create line and paragraph breaks:

‘@*’
Force a line break.

‘@sp N’
Skip N blank lines.

The line-break-prevention command holds text together all on one line:

‘@w{TEXT}’
Prevent TEXT from being split and hyphenated across two lines.

The pagination commands apply only to printed output, since Info files do not have pages.

‘@page’
Start a new page in the printed manual.

`\@group'`

Hold text together that must appear on one printed page.

`\@need MILS'`

Start a new printed page if not enough space on this one.

1.164 texi.guide/Line Breaks

`\@*`: Generate Line Breaks

=====

The `\@*` command forces a line break in both the printed manual and in Info.

For example,

This line `@*` is broken `@*`in two places.

produces

This line
is broken
in two places.

(Note that the space after the first `\@*` command is faithfully carried down to the next line.)

The `\@*` command is often used in a file's copyright page:

This is edition 2.0 of the Texinfo documentation,`@*`
and is for ...

In this case, the `\@*` command keeps TeX from stretching the line across the whole page in an ugly manner.

Please note: Do not write braces after an `\@*` command; they are not needed.

Do not write an `\@refill'` command at the end of a paragraph containing an `\@*` command; it will cause the paragraph to be refilled after the line break occurs, negating the effect of the line break.

1.165 texi.guide/w

`\@w{TEXT}`: Prevent Line Breaks

=====

`\@w{TEXT}'` outputs TEXT and prohibits line breaks within TEXT.

You can use the `'@w'` command to prevent TeX from automatically hyphenating a long name or phrase that accidentally falls near the end of a line.

```
You can copy GNU software from @w{@file{prep.ai.mit.edu}}.
```

produces

```
You can copy GNU software from 'prep.ai.mit.edu'.
```

In the Texinfo file, you must write the `'@w'` command and its argument (all the affected text) all on one line.

**Caution:* Do not write an `'@refill'` command at the end of a paragraph containing an `'@w'` command; it will cause the paragraph to be refilled and may thereby negate the effect of the `'@w'` command.

1.166 texi.guide/sp

`'@sp' N`: Insert Blank Lines

=====

A line beginning with and containing only `'@sp N'` generates N blank lines of space in both the printed manual and the Info file. `'@sp'` also forces a paragraph break. For example,

```
@sp 2
```

generates two blank lines.

The `'@sp'` command is most often used in the title page.

1.167 texi.guide/page

`'@page'`: Start a New Page

=====

A line containing only `'@page'` starts a new page in a printed manual. The command has no effect on Info files since they are not paginated. An `'@page'` command is often used in the `'@titlepage'` section of a Texinfo file to start the copyright page.

1.168 texi.guide/group

`@group': Prevent Page Breaks

=====

The ``@group'` command (on a line by itself) is used inside an ``@example'` or similar construct to begin an unsplittable vertical group, which will appear entirely on one page in the printed output. The group is terminated by a line containing only ``@end group'`. These two lines produce no output of their own, and in the Info file output they have no effect at all.

Although ``@group'` would make sense conceptually in a wide variety of contexts, its current implementation works reliably only within ``@example'` and variants, and within ``@display'`, ``@format'`, ``@flushleft'` and ``@flushright'`. See Quotations and Examples. (What all these commands have in common is that each line of input produces a line of output.) In other contexts, ``@group'` can cause anomalous vertical spacing.

This formatting requirement means that you should write:

```
@example
@group
...
@end group
@end example
```

with the ``@group'` and ``@end group'` commands inside the ``@example'` and ``@end example'` commands.

The ``@group'` command is most often used to hold an example together on one page. In this Texinfo manual, more than 100 examples contain text that is enclosed between ``@group'` and ``@end group'`.

If you forget to end a group, you may get strange and unfathomable error messages when you run TeX. This is because TeX keeps trying to put the rest of the Texinfo file onto the one page and does not start to generate error messages until it has processed considerable text. It is a good rule of thumb to look for a missing ``@end group'` if you get incomprehensible error messages in TeX.

1.169 texi.guide/need**`@need MILS': Prevent Page Breaks**

=====

A line containing only ``@need N'` starts a new page in a printed manual if fewer than N mils (thousandths of an inch) remain on the current page. Do not use braces around the argument N. The ``@need'` command has no effect on Info files since they are not paginated.

This paragraph is preceded by an ``@need'` command that tells TeX to start a new page if fewer than 800 mils (eight-tenths inch) remain on the page. It looks like this:

```
@need 800
```

```
This paragraph is preceded by ...
```

The '@need' command is useful for preventing orphans (single lines at the bottoms of printed pages).

1.170 texi.guide/Definition Commands

Definition Commands

```
*****
```

The '@defn' command and the other "definition commands" enable you to describe functions, variables, macros, commands, user options, special forms and other such artifacts in a uniform format.

In the Info file, a definition causes the entity category--'Function', 'Variable', or whatever--to appear at the beginning of the first line of the definition, followed by the entity's name and arguments. In the printed manual, the command causes TeX to print the entity's name and its arguments on the left margin and print the category next to the right margin. In both output formats, the body of the definition is indented. Also, the name of the entity is entered into the appropriate index: '@defn' enters the name into the index of functions, '@defvr' enters it into the index of variables, and so on.

A manual need not and should not contain more than one definition for a given name. An appendix containing a summary should use '@table' rather than the definition commands.

Def Cmd Template	How to structure a description using a definition command.
Optional Arguments	How to handle optional and repeated arguments.
deffnx	How to group two or more 'first' lines.
Def Cmds in Detail	All the definition commands.
Def Cmd Conventions	Conventions for writing definitions.
Sample Function Definition	

1.171 texi.guide/Def Cmd Template

The Template for a Definition

```
=====
```

The '@defn' command is used for definitions of entities that resemble functions. To write a definition using the '@defn' command, write the '@defn' command at the beginning of a line and follow it on the same line by the category of the entity, the name of the entity itself, and its arguments (if any). Then write the body of the

definition on succeeding lines. (You may embed examples in the body.) Finally, end the definition with an '@end deffn' command written on a line of its own. (The other definition commands follow the same format.)

The template for a definition looks like this:

```
@deffn CATEGORY NAME ARGUMENTS...
BODY-OF-DEFINITION
@end deffn
```

For example,

```
@deffn Command forward-word count
This command moves point forward @var{count} words
(or backward if @var{count} is negative). ...
@end deffn
```

produces

```
- Command: forward-word COUNT
  This function moves point forward COUNT words (or backward if
  COUNT is negative). ...
```

Capitalize the category name like a title. If the name of the category contains spaces, as in the phrase 'Interactive Command', write braces around it. For example:

```
@deffn {Interactive Command} isearch-forward
...
@end deffn
```

Otherwise, the second word will be mistaken for the name of the entity.

Some of the definition commands are more general than others. The '@deffn' command, for example, is the general definition command for functions and the like--for entities that may take arguments. When you use this command, you specify the category to which the entity belongs. The '@deffn' command possesses three predefined, specialized variations, '@defun', '@defmac', and '@defspec', that specify the category for you: "Function", "Macro", and "Special Form" respectively. The '@defvr' command also is accompanied by several predefined, specialized variations for describing particular kinds of variables.

The template for a specialized definition, such as '@defun', is similar to the template for a generalized definition, except that you do not need to specify the category:

```
@defun NAME ARGUMENTS...
BODY-OF-DEFINITION
@end defun
```

Thus,

```
@defun buffer-end flag
This function returns @code{(point-min)} if @var{flag}
is less than 1, @code{(point-max)} otherwise.
```

```
...
@end defun
```

produces

```
- Function: buffer-end FLAG
  This function returns `(point-min)' if FLAG is less than 1,
  `(point-max)' otherwise. ...
```

See Sample Function Definition, for a more detailed example of a function definition, including the use of `@example` inside the definition.

The other specialized commands work like `@defun`.

1.172 texi.guide/Optional Arguments

Optional and Repeated Arguments

Some entities take optional or repeated arguments, which may be specified by a distinctive glyph that uses square brackets and ellipses. For example, a special form often breaks its argument list into separate arguments in more complicated ways than a straightforward function.

An argument enclosed within square brackets is optional. Thus, [OPTIONAL-ARG] means that OPTIONAL-ARG is optional. An argument followed by an ellipsis is optional and may be repeated more than once. Thus, REPEATED-ARGS... stands for zero or more arguments. Parentheses are used when several arguments are grouped into additional levels of list structure in Lisp.

Here is the `@defspec` line of an example of an imaginary special form:

```
- Special Form: foobar (VAR [FROM TO [INC]]) BODY...
```

In this example, the arguments FROM and TO are optional, but must both be present or both absent. If they are present, INC may optionally be specified as well. These arguments are grouped with the argument VAR into a list, to distinguish them from BODY, which includes all remaining elements of the form.

In a Texinfo source file, this `@defspec` line is written like this (except it would not be split over two lines, as it is in this example).

```
@defspec foobar (@var{var} [@var{from} @var{to}
  [@var{inc}]]) @var{body}@dots{}
```

The function is listed in the Command and Variable Index under `'foobar'`.

1.173 texi.guide/defnfx

Two or More 'First' Lines

=====

To create two or more 'first' or header lines for a definition, follow the first '@deffn' line by a line beginning with '@deffnx'. The '@deffnx' command works exactly like '@deffn' except that it does not generate extra vertical white space between it and the preceding line.

For example,

```
@deffn {Interactive Command} isearch-forward
@deffnx {Interactive Command} isearch-backward
These two search commands are similar except ...
@end deffn
```

produces

```
- Interactive Command: isearch-forward
- Interactive Command: isearch-backward
  These two search commands are similar except ...
```

Each of the other definition commands has an 'x' form: '@defunx', '@defvr x', '@deftypefunx', etc.

The 'x' forms work just like '@itemx'; see See '@itemx'.

1.174 texi.guide/Def Cmds in Detail

The Definition Commands

=====

Texinfo provides more than a dozen definition commands, all of which are described in this section.

The definition commands automatically enter the name of the entity in the appropriate index: for example, '@deffn', '@defun', and '@defmac' enter function names in the index of functions; '@defvr' and '@defvar' enter variable names in the index of variables.

Although the examples that follow mostly illustrate Lisp, the commands can be used for other programming languages.

Functions Commands	Commands for functions and similar entities.
Variables Commands	Commands for variables and similar entities.
Typed Functions	Commands for functions in typed languages.
Typed Variables	Commands for variables in typed languages.
Abstract Objects	Commands for object-oriented programming.
Data Types	The definition command for data types.

1.175 texi.guide/Functions Commands

Functions and Similar Entities

This section describes the commands for describing functions and similar entities:

`'@defn CATEGORY NAME ARGUMENTS...'`

The `'@defn'` command is the general definition command for functions, interactive commands, and similar entities that may take arguments. You must choose a term to describe the category of entity being defined; for example, "Function" could be used if the entity is a function. The `'@defn'` command is written at the beginning of a line and is followed on the same line by the category of entity being described, the name of this particular entity, and its arguments, if any. Terminate the definition with `'@end defn'` on a line of its own.

For example, here is a definition:

```
@defn Command forward-char nchars
Move point forward @var{nchars} characters.
@end defn
```

This shows a rather terse definition for a "command" named `'forward-char'` with one argument, `NCHARS`.

`'@defn'` prints argument names such as `NCHARS` in italics or upper case, as if `'@var'` had been used, because we think of these names as metasyntactic variables--they stand for the actual argument values. Within the text of the description, write an argument name explicitly with `'@var'` to refer to the value of the argument. In the example above, we used `'@var{nchars}'` in this way.

The template for `'@defn'` is:

```
@defn CATEGORY NAME ARGUMENTS...
BODY-OF-DEFINITION
@end defn
```

`'@defun NAME ARGUMENTS...'`

The `'@defun'` command is the definition command for functions. `'@defun'` is equivalent to `'@defn Function ...'`.

For example,

```
@defun set symbol new-value
Change the value of the symbol @var{symbol}
to @var{new-value}.
@end defun
```

shows a rather terse definition for a function `'set'` whose arguments are `SYMBOL` and `NEW-VALUE`. The argument names on the `'@defun'` line automatically appear in italics or upper case as if they were enclosed in `'@var'`. Terminate the definition with `'@end`

defun' on a line of its own.

The template is:

```
@defun FUNCTION-NAME ARGUMENTS...
BODY-OF-DEFINITION
@end defun
```

'@defun' creates an entry in the index of functions.

'@defmac NAME ARGUMENTS...'

The '@defmac' command is the definition command for macros.

'@defmac' is equivalent to '@defn Macro ...' and works like '@defun'.

'@defspec NAME ARGUMENTS...'

The '@defspec' command is the definition command for special forms. (In Lisp, a special form is an entity much like a function.) '@defspec' is equivalent to '@defn {Special Form} ...' and works like '@defun'.

1.176 texi.guide/Variables Commands

Variables and Similar Entities

Here are the commands for defining variables and similar entities:

'@defvr CATEGORY NAME'

The '@defvr' command is a general definition command for something like a variable--an entity that records a value. You must choose a term to describe the category of entity being defined; for example, "Variable" could be used if the entity is a variable. Write the '@defvr' command at the beginning of a line and followed it on the same line by the category of the entity and the name of the entity.

Capitalize the category name like a title. If the name of the category contains spaces, as in the name 'User Option', write braces around it. Otherwise, the second word will be mistaken for the name of the entity, for example:

```
@defvr {User Option} fill-column
This buffer-local variable specifies
the maximum width of filled lines.
...
@end defvr
```

Terminate the definition with '@end defvr' on a line of its own.

The template is:

```
@defvr CATEGORY NAME
BODY-OF-DEFINITION
```

```
@end defvr
```

'@defvr' creates an entry in the index of variables for NAME.

```
'@defvar NAME'
```

The '@defvar' command is the definition command for variables.

'@defvar' is equivalent to '@defvr Variable ...'.

For example:

```
@defvar kill-ring
...
@end defvar
```

The template is:

```
@defvar NAME
BODY-OF-DEFINITION
@end defvar
```

'@defvar' creates an entry in the index of variables for NAME.

```
'@defopt NAME'
```

The '@defopt' command is the definition command for user options.

'@defopt' is equivalent to '@defvr {User Option} ...' and works like '@defvar'.

1.177 texi.guide/Typed Functions

Functions in Typed Languages

The '@deftypefn' command and its variations are for describing functions in C or any other language in which you must declare types of variables and functions.

```
'@deftypefn CATEGORY DATA-TYPE NAME ARGUMENTS...'
```

The '@deftypefn' command is the general definition command for functions and similar entities that may take arguments and that are typed. The '@deftypefn' command is written at the beginning of a line and is followed on the same line by the category of entity being described, the type of the returned value, the name of this particular entity, and its arguments, if any.

For example,

```
@deftypefn {Library Function} int foobar
  (int @var{foo}, float @var{bar})
...
@end deftypefn
```

(where the text before the "...", shown above as two lines, would actually be a single line in a real Texinfo file) produces the following in Info:

```
-- Library Function: int foobar (int FOO, float BAR)
...
```

This means that `'foobar'` is a "library function" that returns an `'int'`, and its arguments are `FOO` (an `'int'`) and `BAR` (a `'float'`).

The argument names that you write in `'@deftypefn'` are not subject to an implicit `'@var'`--since the actual names of the arguments in `'@deftypefn'` are typically scattered among data type names and keywords, Texinfo cannot find them without help. Instead, you must write `'@var'` explicitly around the argument names. In the example above, the argument names are `'foo'` and `'bar'`.

The template for `'@deftypefn'` is:

```
@deftypefn CATEGORY DATA-TYPE NAME ARGUMENTS ...
BODY-OF-DESCRIPTION
@end deftypefn
```

Note that if the CATEGORY or DATA TYPE is more than one word then it must be enclosed in braces to make it a single argument.

If you are describing a procedure in a language that has packages, such as Ada, you might consider using `'@deftypefn'` in a manner somewhat contrary to the convention described in the preceding paragraphs.

For example:

```
@deftypefn stacks private push
    (@var{s}:in out stack;
    @var{n}:in integer)
...
@end deftypefn
```

(The `'@deftypefn'` arguments are shown split into three lines, but would be a single line in a real Texinfo file.)

In this instance, the procedure is classified as belonging to the package `'stacks'` rather than classified as a `'procedure'` and its data type is described as `'private'`. (The name of the procedure is `'push'`, and its arguments are `S` and `N`.)

`'@deftypefn'` creates an entry in the index of functions for NAME.

`'@deftypefun DATA-TYPE NAME ARGUMENTS...'`

The `'@deftypefun'` command is the specialized definition command for functions in typed languages. The command is equivalent to `'@deftypefn Function ...'`.

Thus,

```
@deftypefun int foobar (int @var{foo}, float @var{bar})
...
@end deftypefun
```

produces the following in Info:

```
-- Function: int foobar (int FOO, float BAR)
...
```

The template is:

```
@deftypefun TYPE NAME ARGUMENTS...
BODY-OF-DESCRIPTION
@end deftypefun
```

'@deftypefun' creates an entry in the index of functions for NAME.

1.178 texi.guide/Typed Variables

Variables in Typed Languages

Variables in typed languages are handled in a manner similar to functions in typed languages. See Typed Functions. The general definition command '@deftypevr' corresponds to '@deftypefn' and the specialized definition command '@deftypevar' corresponds to '@deftypefun'.

'@deftypevr CATEGORY DATA-TYPE NAME'

The '@deftypevr' command is the general definition command for something like a variable in a typed language--an entity that records a value. You must choose a term to describe the category of the entity being defined; for example, "Variable" could be used if the entity is a variable.

The '@deftypevr' command is written at the beginning of a line and is followed on the same line by the category of the entity being described, the data type, and the name of this particular entity.

For example:

```
@deftypevr {Global Flag} int enable
...
@end deftypevr
```

produces the following in Info:

```
-- Global Flag: int enable
...
```

The template is:

```
@deftypevr CATEGORY DATA-TYPE NAME
BODY-OF-DESCRIPTION
@end deftypevr
```

'@deftypevr' creates an entry in the index of variables for NAME.

`'@deftypevar DATA-TYPE NAME'`

The `'@deftypevar'` command is the specialized definition command for variables in typed languages. `'@deftypevar'` is equivalent to `'@deftypevr Variable ...'`.

For example:

```
@deftypevar int fubar
...
@end deftypevar
```

produces the following in Info:

```
-- Variable: int fubar
...
```

The template is:

```
@deftypevar DATA-TYPE NAME
BODY-OF-DESCRIPTION
@end deftypevar
```

`'@deftypevar'` creates an entry in the index of variables for NAME.

1.179 texi.guide/Abstract Objects

Object-Oriented Programming

Here are the commands for formatting descriptions about abstract objects, such as are used in object-oriented programming. A class is a defined type of abstract object. An instance of a class is a particular object that has the type of the class. An instance variable is a variable that belongs to the class but for which each instance has its own value.

In a definition, if the name of a class is truly a name defined in the programming system for a class, then you should write an `'@code'` around it. Otherwise, it is printed in the usual text font.

`'@defcv CATEGORY CLASS NAME'`

The `'@defcv'` command is the general definition command for variables associated with classes in object-oriented programming. The `'@defcv'` command is followed by three arguments: the category of thing being defined, the class to which it belongs, and its name. Thus,

```
@defcv {Class Option} Window border-pattern
...
@end defcv
```

illustrates how you would write the first line of a definition of the `'border-pattern'` class option of the class `'Window'`.

The template is

```
@defcv CATEGORY CLASS NAME
...
@end defcv
```

'@defcv' creates an entry in the index of variables.

'@defivar CLASS NAME'

The '@defivar' command is the definition command for instance variables in object-oriented programming. '@defivar' is equivalent to '@defcv {Instance Variable} ...'

The template is:

```
@defivar CLASS INSTANCE-VARIABLE-NAME
BODY-OF-DEFINITION
@end defivar
```

'@defivar' creates an entry in the index of variables.

'@defop CATEGORY CLASS NAME ARGUMENTS...'

The '@defop' command is the general definition command for entities that may resemble methods in object-oriented programming. These entities take arguments, as functions do, but are associated with particular classes of objects.

For example, some systems have constructs called "wrappers" that are associated with classes as methods are, but that act more like macros than like functions. You could use '@defop Wrapper' to describe one of these.

Sometimes it is useful to distinguish methods and "operations". You can think of an operation as the specification for a method. Thus, a window system might specify that all window classes have a method named 'expose'; we would say that this window system defines an 'expose' operation on windows in general. Typically, the operation has a name and also specifies the pattern of arguments; all methods that implement the operation must accept the same arguments, since applications that use the operation do so without knowing which method will implement it.

Often it makes more sense to document operations than methods. For example, window application developers need to know about the 'expose' operation, but need not be concerned with whether a given class of windows has its own method to implement this operation. To describe this operation, you would write:

```
@defop Operation windows expose
```

The '@defop' command is written at the beginning of a line and is followed on the same line by the overall name of the category of operation, the name of the class of the operation, the name of the operation, and its arguments, if any.

The template is:

```
@defop CATEGORY CLASS NAME ARGUMENTS...
BODY-OF-DEFINITION
@end defop
```

'@defop' creates an entry, such as ``expose' on 'windows'', in the index of functions.

'@defmethod CLASS NAME ARGUMENTS...'

The '@defmethod' command is the definition command for methods in object-oriented programming. A method is a kind of function that implements an operation for a particular class of objects and its subclasses. In the Lisp Machine, methods actually were functions, but they were usually defined with 'defmethod'.

'@defmethod' is equivalent to '@defop Method ...'. The command is written at the beginning of a line and is followed by the name of the class of the method, the name of the method, and its arguments, if any.

For example,

```
@defmethod 'bar-class' bar-method argument
...
@end defmethod
```

illustrates the definition for a method called 'bar-method' of the class 'bar-class'. The method takes an argument.

The template is:

```
@defmethod CLASS METHOD-NAME ARGUMENTS...
BODY-OF-DEFINITION
@end defmethod
```

'@defmethod' creates an entry in the index of functions, such as ``bar-method' on 'bar-class'.

1.180 texi.guide/Data Types

Data Types

Here is the command for data types:

'@deftp CATEGORY NAME ATTRIBUTES...'

The '@deftp' command is the generic definition command for data types. The command is written at the beginning of a line and is followed on the same line by the category, by the name of the type (which is a word like 'int' or 'float'), and then by names of attributes of objects of that type. Thus, you could use this command for describing 'int' or 'float', in which case you could use 'data type' as the category. (A data type is a category of certain objects for purposes of deciding which operations can be performed on them.)

In Lisp, for example, "pair" names a particular data type, and an object of that type has two slots called the CAR and the CDR. Here is how you would write the first line of a definition of 'pair'.

```
@deftp {Data type} pair car cdr
...
@end deftp
```

The template is:

```
@deftp CATEGORY NAME-OF-TYPE ATTRIBUTES...
BODY-OF-DEFINITION
@end deftp
```

'@deftp' creates an entry in the index of data types.

1.181 texi.guide/Def Cmd Conventions

Conventions for Writing Definitions

=====

When you write a definition using '@deffn', '@defun', or one of the other definition commands, please take care to use arguments that indicate the meaning, as with the COUNT argument to the 'forward-word' function. Also, if the name of an argument contains the name of a type, such as INTEGER, take care that the argument actually is of that type.

1.182 texi.guide/Sample Function Definition

A Sample Function Definition

=====

A function definition uses the '@defun' and '@end defun' commands. The name of the function follows immediately after the '@defun' command and it is followed, on the same line, by the parameter list.

Here is a definition from 'The GNU Emacs Lisp Reference Manual'. (See Calling Functions.)

- Function: apply FUNCTION &rest ARGUMENTS
 'apply' calls FUNCTION with ARGUMENTS, just like 'funcall' but with one difference: the last of ARGUMENTS is a list of arguments to give to FUNCTION, rather than a single argument. We also say that this list is "appended" to the other arguments.
 'apply' returns the result of calling FUNCTION. As with

`'funcall'`, `FUNCTION` must either be a Lisp function or a primitive function; special forms and macros do not make sense in `'apply'`.

```
(setq f 'list)
=> list
(apply f 'x 'y 'z)
error--> Wrong type argument: listp, z
(apply '+ 1 2 '(3 4))
=> 10
(apply '+ '(1 2 3 4))
=> 10

(apply 'append '((a b c) nil (x y z) nil))
=> (a b c x y z)
```

An interesting example of using `'apply'` is found in the description of `'mapcar'`.

In the Texinfo source file, this example looks like this:

```
@defun apply function &rest arguments

@code{apply} calls @var{function} with
@var{arguments}, just like @code{funcall} but with one
difference: the last of @var{arguments} is a list of
arguments to give to @var{function}, rather than a single
argument. We also say that this list is @dfn{appended}
to the other arguments.

@code{apply} returns the result of calling
@var{function}. As with @code{funcall},
@var{function} must either be a Lisp function or a
primitive function; special forms and macros do not make
sense in @code{apply}.

@example
(setq f 'list)
  @result{} list
(apply f 'x 'y 'z)
@error{} Wrong type argument: listp, z
(apply '+ 1 2 '(3 4))
  @result{} 10
(apply '+ '(1 2 3 4))
  @result{} 10

(apply 'append '((a b c) nil (x y z) nil))
  @result{} (a b c x y z)
@end example

An interesting example of using @code{apply} is found
in the description of @code{mapcar}.@refill
@end defun
```

In this manual, this function is listed in the Command and Variable Index under `'apply'`.

Ordinary variables and user options are described using a format like that for functions except that variables do not take arguments.

1.183 texi.guide/Footnotes

Footnotes

A "footnote" is for a reference that documents or elucidates the primary text.(1)

In Texinfo, footnotes are created with the '@footnote' command. This command is followed immediately by a left brace, then by the text of the footnote, and then by a terminating right brace. The template is:

```
@footnote{TEXT}
```

Footnotes may be of any length, but are usually short.

For example, this clause is followed by a sample footnote(2); in the Texinfo source, it looks like this:

```
...a sample footnote @footnote{Here is the sample
footnote.}; in the Texinfo source...
```

In a printed manual or book, the reference mark for a footnote is a small, superscripted number; the text of the footnote is written at the bottom of the page, below a horizontal line.

In Info, the reference mark for a footnote is a pair of parentheses with the footnote number between them, like this: '(1)'.

Info has two footnote styles, which determine where the text of the footnote is located:

- * In the 'End' node style, all the footnotes for a single node are placed at the end of that node. The footnotes are separated from the rest of the node by a line of dashes with the word 'Footnotes' within it. Each footnote begins with an '(N)' reference mark.

Here is an example of a single footnote in the end of node style:

```
----- Footnotes -----
```

```
(1) Here is a sample footnote.
```

- * In the 'Separate' node style, all the footnotes for a single node are placed in an automatically constructed node of their own. In this style, a "footnote reference" follows each '(N)' reference mark in the body of the node. The footnote reference is actually a cross reference which you use to reach the footnote node.

The name of the node containing the footnotes is constructed by

appending `'-Footnotes'` to the name of the node that contains the footnotes. (Consequently, the footnotes' node for the `'Footnotes'` node is `'Footnotes-Footnotes'`!) The footnotes' node has an `'Up'` node pointer that leads back to its parent node.

Here is how the first footnote in this manual looks after being formatted for Info in the separate node style:

```
File: texinfo.info  Node: Overview-Footnotes, Up: Overview
```

```
(1) Note that the first syllable of "Texinfo" is
pronounced like "speck", not "hex". ...
```

A Texinfo file may be formatted into an Info file with either footnote style.

Use the `'@footnotestyle'` command to specify an Info file's footnote style. Write this command at the beginning of a line followed by an argument, either `'end'` for the end node style or `'separate'` for the separate node style.

For example,

```
@footnotestyle end
```

or

```
@footnotestyle separate
```

Write an `'@footnotestyle'` command before or shortly after the end-of-header line at the beginning of a Texinfo file. (If you include the `'@footnotestyle'` command between the start-of-header and end-of-header lines, the region formatting commands will format footnotes as specified.)

If you do not specify a footnote style, the formatting commands use their default style. Currently, `'makeinfo'` uses the `'end'` style, while `'texinfo-format-buffer'` and `'texinfo-format-region'` use the `'separate'` style.

This chapter contains two footnotes.

```
----- Footnotes -----
```

(1) A footnote should complement or expand upon the primary text, but a reader should not need to read a footnote to understand the primary text. For a thorough discussion of footnotes, see `'The Chicago Manual of Style'`, which is published by the University of Chicago Press.

(2) Here is the sample footnote.

1.184 texi.guide/Conditionals

Conditionally Visible Text

```
*****
```

Sometimes it is good to use different text for a printed manual and its corresponding Info file. In this case, you can use the "conditional commands" to specify which text is for the printed manual and which is for the Info file.

Conditional Commands	How to specify text for Info or TeX.
Using Ordinary TeX Commands	You can use any and all TeX commands.
set clear value	How to designate which text to format (for both Info and TeX); and how to set a flag to a string that you can insert.

1.185 texi.guide/Conditional Commands

Using '@ifinfo' and '@iftex'
=====

'@ifinfo' begins segments of text that should be ignored by TeX when it typesets the printed manual. The segment of text appears only in the Info file. The '@ifinfo' command should appear on a line by itself; end the Info-only text with a line containing '@end ifinfo' by itself. At the beginning of a Texinfo file, the Info permissions are contained within a region marked by '@ifinfo' and '@end ifinfo'. (See Info Summary and Permissions.)

The '@iftex' and '@end iftex' commands are similar to the '@ifinfo' and '@end ifinfo' commands, except that they specify text that will appear in the printed manual but not in the Info file.

For example,

```
@iftex
This text will appear only in the printed manual.
@end iftex

@ifinfo
However, this text will appear only in Info.
@end ifinfo
```

The preceding example produces the following line:

However, this text will appear only in Info.

Note how you only see one of the two lines, depending on whether you are reading the Info version or the printed version of this manual.

The '@titlepage' command is a special variant of '@iftex' that is used for making the title and copyright pages of the printed manual. (See '@titlepage'.)

1.186 texi.guide/Using Ordinary TeX Commands

Using Ordinary TeX Commands

=====

Inside a region delineated by `'@iftex'` and `'@end iftex'`, you can embed some PlainTeX commands. Info will ignore these commands since they are only in that part of the file which is seen by TeX. You can write the TeX commands as you would write them in a normal TeX file, except that you must replace the `'\'` used by TeX with an `'@'`. For example, in the `'@titlepage'` section of a Texinfo file, you can use the TeX command `'@vskip'` to format the copyright page. (The `'@titlepage'` command causes Info to ignore the region automatically, as it does with the `'@iftex'` command.)

However, many features of PlainTeX will not work, as they are overridden by features of Texinfo.

You can enter PlainTeX completely, and use `'\'` in the TeX commands, by delineating a region with the `'@tex'` and `'@end tex'` commands. (The `'@tex'` command also causes Info to ignore the region, like the `'@iftex'` command.)

For example, here is a mathematical expression written in PlainTeX:

```
@tex
$$ \chi^2 = \sum_{i=1}^N
      \left( y_i - (a + b x_i)
      \over \sigma_i \right)^2 $$
@end tex
```

The output of this example will appear only in a printed manual. If you are reading this in Info, you will not see anything after this paragraph.

1.187 texi.guide/set clear value

`'@set'`, `'@clear'`, and `'@value'`

=====

You can direct the Texinfo formatting commands to format or ignore parts of a Texinfo file with the `'@set'`, `'@clear'`, `'@ifset'`, and `'@ifclear'` commands.

In addition, you can use the `'@set FLAG'` command to set the value of FLAG to a string of characters; and use `'@value{FLAG}'` to insert that string. You can use `'@set'`, for example, to set a date and use `'@value'` to insert the date in several places in the Texinfo file.

<code>ifset ifclear</code>	Format a region if a flag is set.
<code>value</code>	Replace a flag with a string.
<code>value Example</code>	An easy way to update edition information.

1.188 texi.guide/ifset ifclear

`'@ifset'` and `'@ifclear'`

When a FLAG is set, the Texinfo formatting commands format text between subsequent pairs of `'@ifset FLAG'` and `'@end ifset'` commands. When the FLAG is cleared, the Texinfo formatting commands do *not* format the text.

Use the `'@set FLAG'` command to turn on, or "set", a FLAG; a "flag" can be any single word. The format for the command looks like this:

```
@set FLAG
```

Write the conditionally formatted text between `'@ifset FLAG'` and `'@end ifset'` commands, like this:

```
@ifset FLAG
CONDITIONAL-TEXT
@end ifset
```

For example, you can create one document that has two variants, such as a manual for a `'large'` and `'small'` model:

```
You can use this machine to dig up shrubs
without hurting them.
```

```
@set large
```

```
@ifset large
It can also dig up fully grown trees.
@end ifset
```

```
Remember to replant promptly ...
```

In the example, the formatting commands will format the text between `'@ifset large'` and `'@end ifset'` because the `'large'` flag is set.

Use the `'@clear FLAG'` command to turn off, or "clear", a flag. Clearing a flag is the opposite of setting a flag. The command looks like this:

```
@clear FLAG
```

Write the command on a line of its own.

When FLAG is cleared, the Texinfo formatting commands do *not* format the text between `'@ifset FLAG'` and `'@end ifset'`; that text is ignored and does not appear in either printed or Info output.

For example, if you clear the flag of the preceding example by writing an `'@clear large'` command after the `'@set large'` command (but

before the conditional text), then the Texinfo formatting commands ignore the text between the '@ifset large' and '@end ifset' commands. In the formatted output, that text does not appear; in both printed and Info output, you see only the lines that say, "You can use this machine to dig up shrubs without hurting them. Remember to replant promptly ...".

If a flag is cleared with an '@clear FLAG' command, then the formatting commands format text between subsequent pairs of '@ifclear' and '@end ifclear' commands. But if the flag is set with '@set FLAG', then the formatting commands do *not* format text between an '@ifclear' and an '@end ifclear' command; rather, they ignore that text. An '@ifclear' command looks like this:

```
@ifclear FLAG
```

In brief, the commands are:

'@set FLAG'

Tell the Texinfo formatting commands that FLAG is set.

'@clear FLAG'

Tell the Texinfo formatting commands that FLAG is cleared.

'@ifset FLAG'

If FLAG is set, tell the Texinfo formatting commands to format the text up to the following '@end ifset' command.

If FLAG is cleared, tell the Texinfo formatting commands to ignore text up to the following '@end ifset' command.

'@ifclear FLAG'

If FLAG is set, tell the Texinfo formatting commands to ignore the text up to the following '@end ifclear' command.

If FLAG is cleared, tell the Texinfo formatting commands to format the text up to the following '@end ifclear' command.

1.189 texi.guide/value

'@value'

You can use the '@set' command to specify a value for a flag, which is expanded by the '@value' command. The value is a string of characters.

Write the '@set' command like this:

```
@set foo This is a string.
```

This sets the value of 'foo' to "This is a string."

The Texinfo formatters replace an '@value{FLAG}' command with the

string to which FLAG is set.

Thus, when 'foo' is set as shown above, the Texinfo formatters convert

```
@value{foo}
to
This is a string.
```

You can write an '@value' command within a paragraph; but you must write an '@set' command on a line of its own.

If you write the '@set' command like this:

```
@set foo
```

without specifying a string, the value of 'foo' is an empty string.

If you clear a previously set flag with an '@clear FLAG' command, a subsequent '@value{flag}' command is invalid and the string is replaced with an error message that says '{No value for "FLAG"}'.

For example, if you set 'foo' as follows:

```
@set how-much very, very, very
```

then the formatters transform

```
It is a @value{how-much} wet day.
into
It is a very, very, very wet day.
```

If you write

```
@clear how-much
```

then the formatters transform

```
It is a @value{how-much} wet day.
into
It is a {No value for "how-much"} wet day.
```

1.190 texi.guide/value Example

'@value' Example

You can use the '@value' command to limit the number of places you need to change when you record an update to a manual. Here is how it is done in 'The GNU Make Manual':

Set the flags:

```
@set EDITION 0.35 Beta
```

```
@set VERSION 3.63 Beta
@set UPDATED 14 August 1992
@set UPDATE-MONTH August 1992
```

Write text for the first '@ifinfo' section, for people reading the Texinfo file:

```
This is Edition @value{EDITION},
last updated @value{UPDATED},
of @cite{The GNU Make Manual},
for @code{make}, Version @value{VERSION}.
```

Write text for the title page, for people reading the printed manual:

```
@title GNU Make
@subtitle A Program for Directing Recompilation
@subtitle Edition @value{EDITION}, ...
@subtitle @value{UPDATE-MONTH}
```

(On a printed cover, a date listing the month and the year looks less fussy than a date listing the day as well as the month and year.)

Write text for the Top node, for people reading the Info file:

```
This is Edition @value{EDITION}
of the @cite{GNU Make Manual},
last updated @value{UPDATED}
for @code{make} Version @value{VERSION}.
```

After you format the manual, the text in the first '@ifinfo' section looks like this:

```
This is Edition 0.35 Beta, last updated 14 August 1992,
of 'The GNU Make Manual', for 'make', Version 3.63 Beta.
```

When you update the manual, change only the values of the flags; you do not need to rewrite the three sections.

1.191 texi.guide/Format-Print Hardcopy

Format and Print Hardcopy

There are three major shell commands for making a printed manual from a Texinfo file: one for converting the Texinfo file into a file that will be printed, a second for sorting indices, and a third for printing the formatted document. When you use the shell commands, you can either work directly in the operating system shell or work within a shell inside GNU Emacs.

If you are using GNU Emacs, you can use commands provided by Texinfo mode instead of shell commands. In addition to the three commands to format a file, sort the indices, and print the result, Texinfo mode offers key bindings for commands to recenter the output buffer, show the

print queue, and delete a job from the print queue.

Use TeX	Use TeX to format for hardcopy.
Shell Format & Print	How to format and print a hardcopy manual with shell commands.
Within Emacs	How to format and print from an Emacs shell.
Texinfo Mode Printing	How to format and print in Texinfo mode.
Compile-Command	How to print using Emacs's compile command.
Requirements Summary	TeX formatting requirements summary.
Preparing for TeX	What you need to do to use TeX.
Overfull hboxes	What are and what to do with overfull hboxes.
smallbook	How to print small format books and manuals.
A4 Paper	How to print on European A4 paper.
Cropmarks and Magnification	How to print marks to indicate the size of pages and how to print scaled up output.

1.192 texi.guide/Use TeX

Use TeX
=====

The typesetting program called TeX is used for formatting a Texinfo file. TeX is a very powerful typesetting program and, if used right, does an exceptionally good job. See How to Obtain TeX, for information on how to obtain TeX.

The 'makeinfo', 'texinfo-format-region', and 'texinfo-format-buffer' commands read the very same @-commands in the Texinfo file as does TeX, but process them differently to make an Info file; see See Create an Info File.

1.193 texi.guide/Shell Format & Print

Format and Print Using Shell Commands
=====

Format the Texinfo file with the shell command 'tex' followed by the name of the Texinfo file. This produces a formatted DVI file as well as several auxiliary files containing indices, cross references, etc. The DVI file (for "DeVice Independent" file) can be printed on a wide variety of printers.

The 'tex' formatting command itself does not sort the indices; it writes an output file of unsorted index data. This is a misfeature of TeX. Hence, to generate a printed index, you first need a sorted index to work from. The 'texindex' command sorts indices. (The source file 'texindex.c' comes as part of the standard GNU distribution and is usually installed when Emacs is installed.)

The `'tex'` formatting command outputs unsorted index files under names that obey a standard convention. These names are the name of your main input file to the `'tex'` formatting command, with everything after the first period thrown away, and the two letter names of indices added at the end. For example, the raw index output files for the input file `'foo.texinfo'` would be `'foo.cp'`, `'foo.vr'`, `'foo.fn'`, `'foo.tp'`, `'foo.pg'` and `'foo.ky'`. Those are exactly the arguments to give to `'texindex'`.

Or else, you can use `'??'` as "wild-cards" and give the command in this form:

```
texindex foo.??
```

This command will run `'texindex'` on all the unsorted index files, including any that you have defined yourself using `'@defindex'` or `'@defcodeindex'`. (You may execute `'texindex foo.??'` even if there are similarly named files with two letter extensions that are not index files, such as `'foo.el'`. The `'texindex'` command reports but otherwise ignores such files.)

For each file specified, `'texindex'` generates a sorted index file whose name is made by appending `'s'` to the input file name. The `'@printindex'` command knows to look for a file of that name. `'texindex'` does not alter the raw index output file.

After you have sorted the indices, you need to rerun the `'tex'` formatting command on the Texinfo file. This regenerates a formatted DVI file with up-to-date index entries.(1)

To summarize, this is a three step process:

1. Run the `'tex'` formatting command on the Texinfo file. This generates the formatted DVI file as well as the raw index files with two letter extensions.
2. Run the shell command `'texindex'` on the raw index files to sort them. This creates the corresponding sorted index files.
3. Rerun the `'tex'` formatting command on the Texinfo file. This regenerates a formatted DVI file with the index entries in the correct order. This second run also corrects the page numbers for the cross references. (The tables of contents are always correct.)

You need not run `'texindex'` each time after you run the `'tex'` formatting. If you do not, on the next run, the `'tex'` formatting command will use whatever sorted index files happen to exist from the previous use of `'texindex'`. This is usually OK while you are debugging.

Rather than type the `'tex'` and `'texindex'` commands yourself, you can use `'texi2dvi'`. This shell script is designed to simplify the `'tex'--'texindex'--'tex'` sequence by figuring out whether index files and DVI files are up-to-date. It runs `'texindex'` and `'tex'` only when necessary.

The syntax for `'texi2dvi'` is like this (where `'%'` is the shell prompt):

```
% texi2dvi FILENAME...
```

Finally, you can print the DVI file with the DVI print command. The precise command to use depends on the system; `'lpr -d'` is common. The DVI print command may require a file name without any extension or with a `'.dvi'` extension.

The following commands, for example, sort the indices, format, and print the 'Bison Manual' (where `'%'` is the shell prompt):

```
% tex bison.texinfo
% texindex bison.??
% tex bison.texinfo
% lpr -d bison.dvi
```

(Remember that the shell commands may be different at your site; but these are commonly used versions.)

----- Footnotes -----

(1) If you use more than one index and have cross references to an index other than the first, you must run `'tex'` **three times** to get correct output: once to generate raw index data; again (after `'texindex'`) to output the text of the indices and determine their true page numbers; and a third time to output correct page numbers in cross references to them. However, cross references to indices are rare.

1.194 texi.guide/Within Emacs

From an Emacs Shell ...
=====

You can give formatting and printing commands from a shell within GNU Emacs. To create a shell within Emacs, type `'M-x shell'`. In this shell, you can format and print the document. See *How to Format and Print Using Shell Commands*, for details.

You can switch to and from the shell buffer while `'tex'` is running and do other editing. If you are formatting a long document on a slow machine, this can be very convenient.

You can also use `'texi2dvi'` from an Emacs shell. For example, here is how to use `'texi2dvi'` to format and print `'Using and Porting GNU CC'` from a shell within Emacs (where `'%'` is the shell prompt):

```
% texi2dvi gcc.texinfo
% lpr -d gcc.dvi
```

See *Texinfo Mode Printing*, for more information about formatting and printing in Texinfo mode.

1.195 texi.guide/TeXinfo Mode Printing

Formatting and Printing in TeXinfo Mode

=====

TeXinfo mode provides several predefined key commands for TeX formatting and printing. These include commands for sorting indices, looking at the printer queue, killing the formatting job, and recentering the display of the buffer in which the operations occur.

`'C-c C-t C-r'`

`'M-x texinfo-tex-region'`

Run TeX on the current region.

`'C-c C-t C-b'`

`'M-x texinfo-tex-buffer'`

Run TeX on the current buffer.

`'C-c C-t C-i'`

`'M-x texinfo-texindex'`

Sort the indices of a TeXinfo file that have been formatted with `'texinfo-tex-region'` or `'texinfo-tex-buffer'`.

`'C-c C-t C-p'`

`'M-x texinfo-tex-print'`

Print a DVI file that was made with `'texinfo-tex-region'` or `'texinfo-tex-buffer'`.

`'C-c C-t C-q'`

`'M-x texinfo-show-tex-print-queue'`

Show the print queue.

`'C-c C-t C-d'`

`'M-x texinfo-delete-from-tex-print-queue'`

Delete a job from the print queue; you will be prompted for the job number shown by a preceding `'C-c C-t C-q'` command (`'texinfo-show-tex-print-queue'`).

`'C-c C-t C-k'`

`'M-x texinfo-kill-tex-job'`

Kill either the currently running TeX job that has been started by `'texinfo-tex-region'` or `'texinfo-tex-buffer'`, or any other process running in the TeXinfo shell buffer.

`'C-c C-t C-x'`

`'M-x texinfo-quit-tex-job'`

Quit a TeX formatting job that has stopped because of an error by sending an x to it. When you do this, TeX preserves a record of what it did in a `'.log'` file.

`'C-c C-t C-l'`

`'M-x texinfo-recenter-tex-output-buffer'`

Redisplay the shell buffer in which the TeX printing and formatting commands are run to show its most recent output.

Thus, the usual sequence of commands for formatting a buffer is as

follows (with comments to the right):

C-c C-t C-b	Run TeX on the buffer.
C-c C-t C-i	Sort the indices.
C-c C-t C-b	Rerun TeX to regenerate indices.
C-c C-t C-p	Print the DVI file.
C-c C-t C-q	Display the printer queue.

The Texinfo mode TeX formatting commands start a subshell in Emacs called the `'*texinfo-tex-shell*'`. The `'texinfo-tex-command'`, `'texinfo-texindex-command'`, and `'tex-dvi-print-command'` commands are all run in this shell.

You can watch the commands operate in the `'*texinfo-tex-shell*'` buffer, and you can switch to and from and use the `'*texinfo-tex-shell*'` buffer as you would any other shell buffer.

The formatting and print commands depend on the values of several variables. The default values are:

Variable	Default value
<code>texinfo-tex-command</code>	<code>"tex"</code>
<code>texinfo-texindex-command</code>	<code>"texindex"</code>
<code>texinfo-tex-shell-cd-command</code>	<code>"cd"</code>
<code>texinfo-tex-dvi-print-command</code>	<code>"lpr -d"</code>
<code>texinfo-show-tex-queue-command</code>	<code>"lpq"</code>
<code>texinfo-delete-from-print-queue-command</code>	<code>"lprm"</code>
<code>texinfo-start-of-header</code>	<code>"***start"</code>
<code>texinfo-end-of-header</code>	<code>"***end"</code>
<code>texinfo-tex-trailer</code>	<code>"@bye"</code>

The default values of both the `'texinfo-tex-command'` and the `'texinfo-texindex-command'` variables are set in the `'texinfo-tex.el'` file.

You can change the values of these variables with the `'M-x edit-options'` command (see Editing Variable Values), with the `'M-x set-variable'` command (see Examining and Setting Variables), or with your `'.emacs'` initialization file (see Init File).

1.196 texi.guide/Compile-Command

Using the Local Variables List

=====

Yet another way to apply the TeX formatting command to a Texinfo file is to put that command in a "local variables list" at the end of the Texinfo file. You can then specify the TeX formatting command as a `'compile-command'` and have Emacs run the TeX formatting command by typing `'M-x compile'`. This creates a special shell called the `'*compilation buffer*'` in which Emacs runs the compile command. For example, at the end of the `'gdb.texinfo'` file, after the `'@bye'`, you would put the following:

```
@c Local Variables:
@c compile-command: "tex gdb.texinfo"
@c End:
```

This technique is most often used by programmers who also compile programs this way; see See Compilation.

1.197 texi.guide/Requirements Summary

TeX Formatting Requirements Summary

=====

Every Texinfo file that is to be input to TeX must begin with a `\input` command and contain an `@settitle` command:

```
\input texinfo
@settitle NAME-OF-MANUAL
```

The first command instructs TeX to load the macros it needs to process a Texinfo file and the second command specifies the title of printed manual.

Every Texinfo file must end with a line that terminates TeX processing and forces out unfinished pages:

```
@bye
```

Strictly speaking, these three lines are all a Texinfo file needs for TeX, besides the body. (The `@setfilename` line is the only line that a Texinfo file needs for Info formatting.)

Usually, the file's first line contains an `@c -*-texinfo-*` comment that causes Emacs to switch to Texinfo mode when you edit the file. In addition, the beginning usually includes an `@setfilename` for Info formatting, an `@setchapternewpage` command, a title page, a copyright page, and permissions. Besides an `@bye`, the end of a file usually includes indices and a table of contents.

For more information, see
 See `@setchapternewpage`,
 See Page Headings,
 See Titlepage & Copyright Page,
 See Printing Indices & Menus, and
 See Contents.

1.198 texi.guide/Preparing for TeX

Preparing to Use TeX

=====

TeX needs to know where to find the `'texinfo.tex'` file that you have told it to input with the `'\input texinfo'` command at the beginning of the first line. The `'texinfo.tex'` file tells TeX how to handle distributions.)

Usually, the `'texinfo.tex'` file is put in the default directory that contains TeX macros (the `'/usr/lib/tex/macros'` directory) when GNU Emacs or other GNU software is installed. In this case, TeX will find the file and you do not need to do anything special. Alternatively, you can put `'texinfo.tex'` in the directory in which the Texinfo source file is located, and TeX will find it there.

However, you may want to specify the location of the `'\input'` file yourself. One way to do this is to write the complete path for the file after the `'\input'` command. Another way is to set the `'TEXINPUTS'` environment variable in your `'cshrc'` or `'profile'` file. The `'TEXINPUTS'` environment variable will tell TeX where to find the `'texinfo.tex'` file and any other file that you might want TeX to use.

Whether you use a `'cshrc'` or `'profile'` file depends on whether you use `'csh'`, `'sh'`, or `'bash'` for your shell command interpreter. When you use `'csh'`, it looks to the `'cshrc'` file for initialization information, and when you use `'sh'` or `'bash'`, it looks to the `'profile'` file.

In a `'cshrc'` file, you could use the following `'csh'` command sequence:

```
setenv TEXINPUTS ./usr/me/mylib:/usr/lib/tex/macros
```

In a `'profile'` file, you could use the following `'sh'` command sequence:

```
TEXINPUTS=./usr/me/mylib:/usr/lib/tex/macros
export TEXINPUTS
```

This would cause TeX to look for `'\input'` file first in the current directory, indicated by the `'.'`, then in a hypothetical user's `'me/mylib'` directory, and finally in the system library.

1.199 texi.guide/Overfull hboxes

Overfull "hboxes"
=====

TeX is sometimes unable to typeset a line without extending it into the right margin. This can occur when TeX comes upon what it interprets as a long word that it cannot hyphenate, such as an electronic mail network address or a very long title. When this happens, TeX prints an error message like this:

```
Overfull \hbox (20.76302pt too wide)
```

(In TeX, lines are in "horizontal boxes", hence the term, "hbox". The backslash, `\`, is the TeX equivalent of `@`.)

TeX also provides the line number in the Texinfo source file and the text of the offending line, which is marked at all the places that TeX knows how to hyphenate words. See *Catching Errors with TeX Formatting*, for more information about typesetting errors.

If the Texinfo file has an overfull hbox, you can rewrite the sentence so the overfull hbox does not occur, or you can decide to leave it. A small excursion into the right margin often does not matter and may not even be noticeable.

However, unless told otherwise, TeX will print a large, ugly, black rectangle beside the line that contains the overful hbox. This is so you will notice the location of the problem if you are correcting a draft.

To prevent such a monstrosity from marring your final printout, write the following in the beginning of the Texinfo file on a line of its own, before the `@titlepage` command:

```
@finalout
```

1.200 texi.guide/smallbook

Printing "Small" Books

=====

By default, TeX typesets pages for printing in an 8.5 by 11 inch format. However, you can direct TeX to typeset a document in a 7 by 9.25 inch format that is suitable for bound books by inserting the following command on a line by itself at the beginning of the Texinfo file, before the title page:

```
@smallbook
```

(Since regular sized books are often about 7 by 9.25 inches, this command might better have been called the `@regularbooksize` command, but it came to be called the `@smallbook` command by comparison to the 8.5 by 11 inch format.)

If you write the `@smallbook` command between the start-of-header and end-of-header lines, the Texinfo mode TeX region formatting command, `texinfo-tex-region`, will format the region in "small" book size (see *Start of Header*).

The Free Software Foundation distributes printed copies of 'The GNU Emacs Manual' and other manuals in the "small" book size. See `@smallexample` and `@smalllisp`, for information about commands that make it easier to produce examples for a smaller manual.

1.201 texi.guide/A4 Paper

Printing on A4 Paper

=====

You can tell TeX to typeset a document for printing on European size A4 paper with the '@afourpaper' command. Write the command on a line by itself between '@iftex' and '@end iftex' lines near the beginning of the Texinfo file, before the title page:

For example, this is how you would write the header for this manual:

```
\input texinfo      @c -*-texinfo-*-
@c %**start of header
@setfilename texinfo
@settitle Texinfo
@syncodeindex vr fn
@iftex
@afourpaper
@end iftex
@c %**end of header
```

1.202 texi.guide/Cropmarks and Magnification

Cropmarks and Magnification

=====

You can attempt to direct TeX to print cropmarks at the corners of pages with the '@cropmarks' command. Write the '@cropmarks' command on a line by itself between '@iftex' and '@end iftex' lines near the beginning of the Texinfo file, before the title page, like this:

```
@iftex
@cropmarks
@end iftex
```

This command is mainly for printers that typeset several pages on one sheet of film; but you can attempt to use it to mark the corners of a book set to 7 by 9.25 inches with the '@smallbook' command. (Printers will not produce cropmarks for regular sized output that is printed on regular sized paper.) Since different printing machines work in different ways, you should explore the use of this command with a spirit of adventure. You may have to redefine the command in the 'texinfo.tex' definitions file.

You can attempt to direct TeX to typeset pages larger or smaller than usual with the '\mag' TeX command. Everything that is typeset is scaled proportionally larger or smaller. ('\mag' stands for "magnification".) This is *not* a Texinfo @-command, but is a PlainTeX command that is prefixed with a backslash. You have to write this command between '@tex' and '@end tex' (see Using Ordinary TeX Commands).

Follow the '\mag' command with an '=' and then a number that is 1000

times the magnification you desire. For example, to print pages at 1.2 normal size, write the following near the beginning of the Texinfo file, before the title page:

```
@tex
\mag=1200
@end tex
```

With some printing technologies, you can print normal-sized copies that look better than usual by using a larger-than-normal master.

Depending on your system, `\mag` may not work or may work only at certain magnifications. Be prepared to experiment.

1.203 texi.guide/Create an Info File

Creating an Info 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.

A Texinfo file must possess an `'@setfilename'` line near its beginning, otherwise the Info formatting commands will fail.

For information on installing the Info file in the Info system, see See Install an Info File.

makeinfo advantages	<code>'makeinfo'</code> provides better error checking.
Invoking makeinfo	How to run <code>'makeinfo'</code> from a shell.
makeinfo options	Specify fill-column and other options.
Pointer Validation	How to check that pointers point somewhere.
makeinfo in Emacs	How to run <code>'makeinfo'</code> from Emacs.
texinfo-format commands	Two Info formatting commands written in Emacs Lisp are an alternative to <code>'makeinfo'</code> .
Batch Formatting	How to format for Info in Emacs Batch mode.
Tag and Split Files	How tagged and split files help Info to run better.

1.204 texi.guide/makeinfo advantages

`'makeinfo'` Preferred

=====

The `'makeinfo'` utility creates an Info file from a Texinfo source file more quickly than either of the Emacs formatting commands and provides better error messages. We recommend it. `'makeinfo'` is a C

program that is independent of Emacs. You do not need to run Emacs to use `'makeinfo'`, which means you can use `'makeinfo'` on machines that are too small to run Emacs. You can run `'makeinfo'` in any one of three ways: from an operating system shell, from a shell inside Emacs, or by typing a key command in Texinfo mode in Emacs.

The `'texinfo-format-region'` and the `'texinfo-format-buffer'` commands are useful if you cannot run `'makeinfo'`. Also, in some circumstances, they format short regions or buffers more quickly than `'makeinfo'`.

1.205 texi.guide/Invoking makeinfo

Invoking `'makeinfo'` from a Shell
=====

To create an Info file from a Texinfo file, type `'makeinfo'` followed by the name of the Texinfo file. Thus, to create the Info file for Bison, type the following at the shell prompt (where `'%'` is the prompt):

```
% makeinfo bison.texinfo
```

(You can run a shell inside Emacs by typing `'M-x shell'`.)

Sometimes you will want to specify options. For example, if you wish to discover which version of `'makeinfo'` you are using, type:

```
% makeinfo --version
```

See `makeinfo` options, for more information.

1.206 texi.guide/makeinfo options

Options for `'makeinfo'`
=====

The `'makeinfo'` command takes a number of options. Most often, options are used to set the value of the fill column and specify the footnote style. Each command line option is a word preceded by `'--'` (1) or a letter preceded by `'-'`. You can use abbreviations for the option names as long as they are unique.

For example, you could use the following command to create an Info file for `'bison.texinfo'` in which each line is filled to only 68 columns (where `'%'` is the prompt):

```
% makeinfo --fill-column=68 bison.texinfo
```

You can write two or more options in sequence, like this:

```
% makeinfo --no-split --fill-column=70 ...
```

This would keep the Info file together as one possibly very long file and would also set the fill column to 70.

The options are:

`'-D VAR'`

Cause VAR to be defined. This is equivalent to `'@set VAR'` in the Texinfo file.

`'--error-limit LIMIT'`

Set the maximum number of errors that `'makeinfo'` will report before exiting (on the assumption that continuing would be useless). The default number of errors that can be reported before `'makeinfo'` gives up is 100.

`'--fill-column WIDTH'`

Specify the maximum number of columns in a line; this is the right-hand edge of a line. Paragraphs that are filled will be filled to this width. (Filling is the process of breaking up and connecting lines so that lines are the same length as or shorter than the number specified as the fill column. Lines are broken between words.) The default value for `'fill-column'` is 72.

`'--footnote-style STYLE'`

Set the footnote style to STYLE, either `'end'` for the end node style or `'separate'` for the separate node style. The value set by this option overrides the value set in a Texinfo file by an `'@footnotestyle'` command. When the footnote style is `'separate'`, `'makeinfo'` makes a new node containing the footnotes found in the current node. When the footnote style is `'end'`, `'makeinfo'` places the footnote references at the end of the current node.

`'-I DIR'`

Add `'dir'` to the directory search list for finding files that are included using the `'@include'` command. By default, `'makeinfo'` searches only the current directory.

`'--no-headers'`

Do not include menus or node lines in the output. This results in an ASCII file that you cannot read in Info since it does not contain the requisite nodes or menus; but you can print such a file in a single, typewriter-like font and produce acceptable output.

`'--no-split'`

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

`'--no-pointer-validate'`

`'--no-validate'`

Suppress the pointer-validation phase of `'makeinfo'`. Normally, after a Texinfo file is processed, some consistency checks are made to ensure that cross references can be resolved, etc. See Pointer Validation.

`--no-warn'`

Suppress the output of warning messages. This does *not* suppress the output of error messages, only warnings. You might want this if the file you are creating has examples of Texinfo cross references within it, and the nodes that are referenced do not actually exist.

`--no-number-footnotes'`

Suppress automatic footnote numbering. By default, `makeinfo` numbers 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 and not to 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 INDENT'`

Set the paragraph indentation style to INDENT. The value set by this option overrides the value set in a Texinfo file by an `@paragraphindent` command. The value of INDENT is interpreted as follows:

- * If the value of INDENT is `'asis'`, do not change the existing indentation at the starts of paragraphs.
- * If the value of INDENT is zero, delete any existing indentation.
- * If the value of INDENT is greater than zero, indent each paragraph by that number of spaces.

`--reference-limit LIMIT'`

Set the value of the number of references to a node that `makeinfo` will make without reporting a warning. If a node has more than this number of references in it, `makeinfo` will make the references but also report a warning.

`-U VAR'`

Cause VAR to be undefined. This is equivalent to `@clear VAR` in the Texinfo file.

`--verbose'`

Cause `makeinfo` to display messages saying what it is doing. Normally, `makeinfo` only outputs messages if there are errors or warnings.

`--version'`

Report the version number of this copy of `makeinfo`.

----- Footnotes -----

(1) `--` has replaced `+`, the old introductory character, to maintain POSIX.2 compatibility without losing long-named options.

1.207 texi.guide/Pointer Validation

Pointer Validation
=====

'makeinfo' will check the validity of the final Info file unless you suppress pointer-validation by using the '--no-pointer-validation' option. Mostly, this means ensuring that nodes you have referenced really exist. Here is a complete list of what is checked:

1. If a 'Next', 'Previous', or 'Up' node reference is a reference to a node in the current file and is not an external reference such as to '(dir)', then the referenced node must exist.
2. In every node, if the 'Previous' node is different from the 'Up' node, then the 'Previous' node must also be pointed to by a 'Next' node.
3. Every node except the 'Top' node must have an 'Up' pointer.
4. The node referenced by an 'Up' pointer must contain a reference to the current node in some manner other than through a 'Next' reference. This includes menu entries and cross references.
5. If the 'Next' reference of a node is not the same as the 'Next' reference of the 'Up' reference, then the node referenced by the 'Next' pointer must have a 'Previous' pointer that points back to the current node. This rule allows the last node in a section to point to the first node of the next chapter.

1.208 texi.guide/makeinfo in Emacs

Running 'makeinfo' inside Emacs
=====

You can run 'makeinfo' in GNU Emacs Texinfo mode by using either the 'makeinfo-region' or the 'makeinfo-buffer' commands. In Texinfo mode, the commands are bound to 'C-c C-m C-r' and 'C-c C-m C-b' by default.

```
'C-c C-m C-r'
'M-x makeinfo-region'
    Format the current region for Info.
```

```
'C-c C-m C-b'
'M-x makeinfo-buffer'
    Format the current buffer for Info.
```

When you invoke either 'makeinfo-region' or 'makeinfo-buffer', Emacs prompts for a file name, offering the name of the visited file as the

default. You can edit the default file name in the minibuffer if you wish, before typing RET to start the 'makeinfo' process.

The Emacs 'makeinfo-region' and 'makeinfo-buffer' commands run the 'makeinfo' program in a temporary shell buffer. If 'makeinfo' finds any errors, Emacs displays the error messages in the temporary buffer.

You can parse the error messages by typing 'C-x ``' ('next-error'). This causes Emacs to go to and position the cursor on the line in the Texinfo source that 'makeinfo' thinks caused the error. See Running 'make' or Compilers Generally, for more information about using the 'next-error' command.

In addition, you can kill the shell in which the 'makeinfo' command is running or make the shell buffer display its most recent output.

'C-c C-m C-k'

'M-x makeinfo-kill-job'

Kill the currently running job created by 'makeinfo-region' or 'makeinfo-buffer'.

'C-c C-m C-l'

'M-x makeinfo-recenter-output-buffer'

Redisplay the 'makeinfo' shell buffer to display its most recent output.

(Note that the parallel commands for killing and recentering a TeX job are 'C-c C-t C-k' and 'C-c C-t C-l'. See Texinfo Mode Printing.)

You can specify options for 'makeinfo' by setting the 'makeinfo-options' variable with either the 'M-x edit-options' or the 'M-x set-variable' command, or by setting the variable in your '.emacs' initialization file.

For example, you could write the following in your '.emacs' file:

```
(setq makeinfo-options
  "--paragraph-indent=0 --no-split
  --fill-column=70 --verbose")
```

For more information, see
 See Editing Variable Values,
 See Examining and Setting Variables,
 See Init File, and
 See Options for 'makeinfo'.

1.209 texi.guide/texinfo-format commands

The 'texinfo-format...' Commands

=====

In GNU Emacs in Texinfo mode, you can format part or all of a Texinfo file with the 'texinfo-format-region' command. This formats the current region and displays the formatted text in a temporary buffer

called `*Info Region*`.

Similarly, you can format a buffer with the `'texinfo-format-buffer'` command. This command creates a new buffer and generates the Info file in it. Typing `'C-x C-s'` will save the Info file under the name specified by the `'@setfilename'` line which must be near the beginning of the Texinfo file.

```
'C-c C-e C-r'
''texinfo-format-region''
    Format the current region for Info.
```

```
'C-c C-e C-b'
''texinfo-format-buffer''
    Format the current buffer for Info.
```

The `'texinfo-format-region'` and `'texinfo-format-buffer'` commands provide you with some error checking, and other functions can provide you with further help in finding formatting errors. These procedures are described in an appendix; see *See Catching Mistakes*. However, the `'makeinfo'` program is often faster and provides better error checking (see `makeinfo` in Emacs).

1.210 texi.guide/Batch Formatting

Batch Formatting
=====

You can format Texinfo files for Info using `'batch-texinfo-format'` and Emacs Batch mode. You can run Emacs in Batch mode from any shell, including a shell inside of Emacs. (See *Command Line Switches and Arguments*.)

Here is the command to format all the files that end in `'.texinfo'` in the current directory (where `'%'` is the shell prompt):

```
% emacs -batch -funcall batch-texinfo-format *.texinfo
```

Emacs processes all the files listed on the command line, even if an error occurs while attempting to format some of them.

Run `'batch-texinfo-format'` only with Emacs in Batch mode as shown; it is not interactive. It kills the Batch mode Emacs on completion.

`'batch-texinfo-format'` is convenient if you lack `'makeinfo'` and want to format several Texinfo files at once. When you use Batch mode, you create a new Emacs process. This frees your current Emacs, so you can continue working in it. (When you run `'texinfo-format-region'` or `'texinfo-format-buffer'`, you cannot use that Emacs for anything else until the command finishes.)

1.211 texi.guide/Tag and Split Files

Tag Files and Split Files

=====

If a Texinfo file has more than 30,000 bytes, `'texinfo-format-buffer'` automatically creates a tag table for its Info file; `'makeinfo'` always creates a tag table. With a "tag table", Info can jump to new nodes more quickly than it can otherwise.

In addition, if the Texinfo file contains more than about 70,000 bytes, `'texinfo-format-buffer'` and `'makeinfo'` split the large Info file into shorter "indirect" subfiles of about 50,000 bytes each. Big files are split into smaller files so that Emacs does not need to make a large buffer to hold the whole of a large Info file; instead, Emacs allocates just enough memory for the small, split off file that is needed at the time. This way, Emacs avoids wasting memory when you run Info. (Before splitting was implemented, Info files were always kept short and "include files" were designed as a way to create a single, large printed manual out of the smaller Info files. See Include Files, for more information. Include files are still used for very large documents, such as `'The Emacs Lisp Reference Manual'`, in which each chapter is a separate file.)

When a file is split, Info itself makes use of a shortened version of the original file that contains just the tag table and references to the files that were split off. The split off files are called "indirect" files.

The split off files have names that are created by appending `'-1'`, `'-2'`, `'-3'` and so on to the file name specified by the `'@setfilename'` command. The shortened version of the original file continues to have the name specified by `'@setfilename'`.

At one stage in writing this document, for example, the Info file was saved as `'test-texinfo'` and that file looked like this:

```
Info file: test-texinfo,    -*-Text-*-
produced by texinfo-format-buffer
from file: new-texinfo-manual.texinfo
```

```
^_
Indirect:
test-texinfo-1: 102
test-texinfo-2: 50422
test-texinfo-3: 101300
^_^L
Tag table:
(Indirect)
Node: overview^?104
Node: info file^?1271
Node: printed manual^?4853
Node: conventions^?6855
...
```

(But `'test-texinfo'` had far more nodes than are shown here.) Each of

the split off, indirect files, `'test-texinfo-1'`, `'test-texinfo-2'`, and `'test-texinfo-3'`, is listed in this file after the line that says `'Indirect:'`. The tag table is listed after the line that says `'Tag table:'`.

In the list of indirect files, the number following the file name records the cumulative number of bytes in the preceding indirect files, not counting the file list itself, the tag table, or the permissions text in each file. In the tag table, the number following the node name records the location of the beginning of the node, in bytes from the beginning.

If you are using `'texinfo-format-buffer'` to create Info files, you may want to run the `'Info-validate'` command. (The `'makeinfo'` command does such a good job on its own, you do not need `'Info-validate'`.) However, you cannot run the `'M-x Info-validate'` node-checking command on indirect files. For information on how to prevent files from being split and how to validate the structure of the nodes, see See Using Info-validate.

1.212 texi.guide/Install an Info File

Installing an Info File

Info files are usually kept in the `'info'` directory. (You can find the location of this directory within Emacs by typing `'C-h i'` to enter Info and then typing `'C-x C-f'` to see the full pathname to the `'info'` directory.)

Directory file	The top level menu for all Info files.
New Info File	Listing a new info file.
Other Info Directories	How to specify Info files that are located in other directories.

1.213 texi.guide/Directory file

The `'dir'` File

=====

For Info to work, the `'info'` directory must contain a file that serves as a top level directory for the Info system. By convention, this file is called `'dir'`. The `'dir'` file is itself an Info file. It contains the top level menu for all the Info files in the system. The menu looks like this:

```
* Menu:

* Info:      (info).      Documentation browsing system.
```

```

* Emacs:  (emacs).      The extensible, self-documenting
                        text editor.
* Texinfo: (texinfo).   With one source file, make
                        either a printed manual using
                        TeX or an Info file.
...

```

Each of these menu entries points to the 'Top' node of the Info file that is named in parentheses. (The menu entry does not need to specify the 'Top' node, since Info goes to the 'Top' node if no node name is mentioned. See Nodes in Other Info Files.)

Thus, the 'Info' entry points to the 'Top' node of the 'info' file and the 'Emacs' entry points to the 'Top' node of the 'emacs' file.

In each of the Info files, the 'Up' pointer of the 'Top' node refers back to the 'dir' file. For example, the line for the 'Top' node of the Emacs manual looks like this in Info:

```
File: emacs  Node: Top, Up: (DIR), Next: Distrib
```

(Note that in this case, the 'dir' file name is written in upper case letters--it can be written in either upper or lower case. Info has a feature that it will change the case of the file name to lower case if it cannot find the name as written.)

1.214 texi.guide/New Info File

Listing a New Info File

=====

To add a new Info file to your system, write a menu entry for it in the menu in the 'dir' file in the 'info' directory. Also, move the new Info file itself to the 'info' directory. For example, if you were adding documentation for GDB, you would write the following new entry:

```
* GDB: (gdb).          The source-level C debugger.
```

The first part of the menu entry is the menu entry name, followed by a colon. The second part is the name of the Info file, in parentheses, followed by a period. The third part is the description.

Conventionally, the name of an Info file has a '.info' extension. Thus, you might list the name of the file like this:

```
* GDB: (gdb.info).     The source-level C debugger.
```

However, Info will look for a file with a '.info' extension if it does not find the file under the name given in the menu. This means that you can refer to the file 'gdb.info' as 'gdb', as shown in the first example. This looks better.

1.215 texi.guide/Other Info Directories

Info Files in Other Directories

=====

If an Info file is not in the 'info' directory, there are two ways to specify its location:

- * Write the pathname as the menu's second part, or;
- * Specify the 'info' directory name in an environment variable in your '.profile' or '.cshrc' initialization file. (Only you and others with the same environment variable will be able to find Info files whose location is specified this way.)

For example, to reach a test file in the '~bob/manuals' directory, you could add an entry like this to the menu in the 'dir' file:

- * Test: (~bob/manuals/info-test). Bob's own test file.

In this case, the absolute file name of the 'info-test' file is written as the second part of the menu entry.

Alternatively, you can tell Info where to look by setting the 'INFOPATH' environment variable in your '.cshrc' or '.profile' file.

If you use 'sh' or 'bash' for your shell command interpreter, you must set the 'INFOPATH' environment variable in the '.profile' initialization file; but if you use 'csh', you must set the variable in the '.cshrc' initialization file. The two files require slightly different command formats.

- * In a '.cshrc' file, you could set the 'INFOPATH' variable as follows:

```
setenv INFOPATH .:~bob/manuals:/gnu/emacs/info
```

- * In a '.profile' file, you would achieve the same effect by writing:

```
INFOPATH=.:~bob/manuals:/gnu/emacs/info
export INFOPATH
```

Either form would cause Info to look first in the current directory, indicated by the '.', then in the '~bob/manuals' directory, and finally in the '/gnu/emacs/info' directory (which is a common location for the standard Info directory).

1.216 texi.guide/Command List

Here is an alphabetical list of the @-commands in Texinfo. Square brackets, [], indicate optional arguments; an ellipsis, '...',

indicates repeated text.

`\@*`

Force a line break. Do not end a paragraph that uses `\@*` with an `\@refill` command. See Line Breaks.

`\@.`

Stands for a period that really does end a sentence (usually after an end-of-sentence capital letter). See Controlling Spacing.

`\@:`

Indicate to TeX that an immediately preceding period, question mark, exclamation mark, or colon does not end a sentence. Prevent TeX from inserting extra whitespace as it does at the end of a sentence. The command has no effect on the Info file output. See Controlling Spacing.

`\@@`

Stands for `\@`. See Inserting `\@`.

`\`

Stands for a left-hand brace, `\{`. See Inserting @ braces and periods.

`\@}`

Stands for a right-hand brace, `\}`. See Inserting @ braces and periods.

`\@appendix TITLE`

Begin an appendix. The title appears in the table of contents of a printed manual. In Info, the title is underlined with asterisks. See The `\@unnumbered` and `\@appendix` Commands.

`\@appendixsec TITLE`

`\@appendixsection TITLE`

Begin an appendix section within an appendix. The section title appears in the table of contents of a printed manual. In Info, the title is underlined with equal signs. `\@appendixsection` is a longer spelling of the `\@appendixsec` command. See Section Commands.

`\@appendixsubsec TITLE`

Begin an appendix subsection within an appendix. The title appears in the table of contents of a printed manual. In Info, the title is underlined with hyphens. See Subsection Commands.

`\@appendixsubsubsec TITLE`

Begin an appendix subsubsection within a subappendix. The title appears in the table of contents of a printed manual. In Info, the title is underlined with periods. See The `\subsub` Commands.

`\@asis`

Used following `\@table`, `\@ftable`, and `\@vtable` to print the table's first column without highlighting ("as is"). See Making a Two-column Table.

`\@author AUTHOR`

Typeset AUTHOR flushleft and underline it. See The '@title' and '@author' Commands.

'@b{TEXT}'

Print TEXT in bold font. No effect in Info. See Fonts.

'@bullet{'}

Generate a large round dot, or the closest possible thing to one. See '@bullet'.

'@bye'

Stop formatting a file. The formatters do not see the contents of a file following an '@bye' command. See Ending a File.

'@c COMMENT'

Begin a comment in Texinfo. The rest of the line does not appear in either the Info file or the printed manual. A synonym for '@comment'. See General Syntactic Conventions.

'@cartouche'

Highlight an example or quotation by drawing a box with rounded corners around it. Pair with '@end cartouche'. No effect in Info. See Drawing Cartouches Around Examples.)

'@center LINE-OF-TEXT'

Center the line of text following the command. See '@center'.

'@chapheading TITLE'

Print a chapter-like heading in the text, but not in the table of contents of a printed manual. In Info, the title is underlined with asterisks. See '@majorheading' and '@chapheading'.

'@chapter TITLE'

Begin a chapter. The chapter title appears in the table of contents of a printed manual. In Info, the title is underlined with asterisks. See '@chapter'.

'@cindex ENTRY'

Add ENTRY to the index of concepts. See Defining the Entries of an Index.

'@cite{REFERENCE}'

Highlight the name of a book or other reference that lacks a companion Info file. See '@cite'.

'@clear FLAG'

Unset FLAG, preventing the Texinfo formatting commands from formatting text between subsequent pairs of '@ifset FLAG' and '@end ifset' commands, and preventing '@value{FLAG}' from expanding to the value to which FLAG is set. See '@set' '@clear' '@value'.

'@code{SAMPLE-CODE}'

Highlight text that is an expression, a syntactically complete token of a program, or a program name. See '@code'.

'@comment COMMENT'

Begin a comment in Texinfo. The rest of the line does not appear in either the Info file or the printed manual. A synonym for '@c'. See General Syntactic Conventions.

'@contents'

Print a complete table of contents. Has no effect in Info, which uses menus instead. See Generating a Table of Contents.

'@copyright{'

Generate a copyright symbol. See '@copyright'.

'@defcodeindex INDEX-NAME'

Define a new index and its indexing command. Print entries in an '@code' font. See Defining New Indices.

'@defcv CATEGORY CLASS NAME'

Format a description for a variable associated with a class in object-oriented programming. Takes three arguments: the category of thing being defined, the class to which it belongs, and its name. See Definition Commands.

'@deffn CATEGORY NAME ARGUMENTS...'

Format a description for a function, interactive command, or similar entity that may take arguments. '@deffn' takes as arguments the category of entity being described, the name of this particular entity, and its arguments, if any. See Definition Commands.

'@defindex INDEX-NAME'

Define a new index and its indexing command. Print entries in a roman font. See Defining New Indices.

'@defivar CLASS INSTANCE-VARIABLE-NAME'

Format a description for an instance variable in object-oriented programming. The command is equivalent to '@defcv {Instance Variable} ...'. See Definition Commands.

'@defmac MACRO-NAME ARGUMENTS...'

Format a description for a macro. The command is equivalent to '@deffn Macro ...'. See Definition Commands.

'@defmethod CLASS METHOD-NAME ARGUMENTS...'

Format a description for a method in object-oriented programming. The command is equivalent to '@defop Method ...'. Takes as arguments the name of the class of the method, the name of the method, and its arguments, if any. See Definition Commands.

'@defop CATEGORY CLASS NAME ARGUMENTS...'

Format a description for an operation in object-oriented programming. '@defop' takes as arguments the overall name of the category of operation, the name of the class of the operation, the name of the operation, and its arguments, if any. See Definition Commands.

'@defopt OPTION-NAME'

Format a description for a user option. The command is equivalent to '@defvr {User Option} ...'. See Definition Commands.

`'@defspec SPECIAL-FORM-NAME ARGUMENTS...'`

Format a description for a special form. The command is equivalent to `'@deffn {Special Form} ...'`. See Definition Commands.

`'@deftp CATEGORY NAME-OF-TYPE ATTRIBUTES...'`

Format a description for a data type. `'@deftp'` takes as arguments the category, the name of the type (which is a word like `'int'` or `'float'`), and then the names of attributes of objects of that type. See Definition Commands.

`'@deftypefn CLASSIFICATION DATA-TYPE NAME ARGUMENTS...'`

Format a description for a function or similar entity that may take arguments and that is typed. `'@deftypefn'` takes as arguments the classification of entity being described, the type, the name of the entity, and its arguments, if any. See Definition Commands.

`'@deftypefun DATA-TYPE FUNCTION-NAME ARGUMENTS...'`

Format a description for a function in a typed language. The command is equivalent to `'@deftypefn Function ...'`. See Definition Commands.

`'@deftypevr CLASSIFICATION DATA-TYPE NAME'`

Format a description for something like a variable in a typed language--an entity that records a value. Takes as arguments the classification of entity being described, the type, and the name of the entity. See Definition Commands.

`'@deftypevar DATA-TYPE VARIABLE-NAME'`

Format a description for a variable in a typed language. The command is equivalent to `'@deftypevr Variable ...'`. See Definition Commands.

`'@defun FUNCTION-NAME ARGUMENTS...'`

Format a description for functions. The command is equivalent to `'@deffn Function ...'`. See Definition Commands.

`'@defvar VARIABLE-NAME'`

Format a description for variables. The command is equivalent to `'@defvr Variable ...'`. See Definition Commands.

`'@defvr CATEGORY NAME'`

Format a description for any kind of variable. `'@defvr'` takes as arguments the category of the entity and the name of the entity. See Definition Commands.

`'@dfn{TERM}'`

Highlight the introductory or defining use of a term. See `'@dfn'`.

`'@display'`

Begin a kind of example. Indent text, do not fill, do not select a new font. Pair with `'@end display'`. See `'@display'`.

`'@dmn{DIMENSION}'`

Format a dimension. Cause TeX to insert a narrow space before DIMENSION. No effect in Info. Use for writing a number followed

by an abbreviation of a dimension name, such as `'12pt'`, written as `'12@dmn{pt}'`, with no space between the number and the `'@dmn'` command. See `'@dmn'`.

`'@dots{'`

Insert an ellipsis: `'...'`. See `'@dots'`.

`'@emph{TEXT}'`

Highlight TEXT; text is displayed in *italics* in printed output, and surrounded by asterisks in Info. See Emphasizing Text.

`'@enumerate [NUMBER-OR-LETTER]'`

Begin a numbered list, using `'@item'` for each entry. Optionally, start list with NUMBER-OR-LETTER. Pair with `'@end enumerate'`. See `'@enumerate'`.

`'@equiv{'`

Indicate to the reader the exact equivalence of two forms with a glyph: `'=='`. See Equivalence.

`'@error{'`

Indicate to the reader with a glyph that the following text is an error message: `'error-->'`. See Error Glyph.

`'@evenfooting [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page footings for even-numbered (left-hand) pages. Not relevant to Info. See How to Make Your Own Headings.

`'@evenheading [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page headings for even-numbered (left-hand) pages. Not relevant to Info. See How to Make Your Own Headings.

`'@everyfooting [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page footings for every page. Not relevant to Info. See How to Make Your Own Headings.

`'@everyheading [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page headings for every page. Not relevant to Info. See How to Make Your Own Headings.

`'@example'`

Begin an example. Indent text, do not fill, and select fixed-width font. Pair with `'@end example'`. See `'@example'`.

`'@exdent LINE-OF-TEXT'`

Remove any indentation a line might have. See Undoing the Indentation of a Line.

`'@expansion{'`

Indicate the result of a macro expansion to the reader with a special glyph: `'==>'`. See `'==>'` Indicating an Expansion.

`'@file{FILENAME}'`

Highlight the name of a file, buffer, node, or directory. See `'@file'`.

`'@finalout'`

Prevent TeX from printing large black warning rectangles beside over-wide lines. See `Overfull hboxes`.

`\@findex ENTRY`

Add ENTRY to the index of functions. See `Defining the Entries of an Index`.

`\@flushleft`

Left justify every line but leave the right end ragged. Leave font as is. Pair with `\@end flushleft`. See `\@flushleft` and `\@flushright`.

`\@flushright`

Right justify every line but leave the left end ragged. Leave font as is. Pair with `\@end flushright`. See `\@flushleft` and `\@flushright`.

`\@footnote{TEXT-OF-FOOTNOTE}`

Enter a footnote. Footnote text is printed at the bottom of the page by TeX; Info may format in either `'End'` node or `'Separate'` node style. See `Footnotes`.

`\@footnotestyle STYLE`

Specify an Info file's footnote style, either `'end'` for the end node style or `'separate'` for the separate node style. See `Footnotes`.

`\@format`

Begin a kind of example. Like `\@example` or `\@display`, but do not narrow the margins and do not select the fixed-width font. Pair with `\@end format`. See `\@example`.

`\@ftable FORMATTING-COMMAND`

Begin a two-column table, using `\@item` for each entry. Automatically enter each of the items in the first column into the index of functions. Pair with `\@end ftable`. The same as `\@table`, except for indexing. See `\@ftable` and `\@vtable`.

`\@group`

Hold text together that must appear on one printed page. Pair with `\@end group`. Not relevant to Info. See `\@group`.

`\@heading TITLE`

Print an unnumbered section-like heading in the text, but not in the table of contents of a printed manual. In Info, the title is underlined with equal signs. See `Section Commands`.

`\@headings ON-OFF-SINGLE-DOUBLE`

Turn page headings on or off, or specify single-sided or double-sided page headings for printing. `\@headings on` is synonymous with `\@headings double`. See The `\@headings` Command.

`\@i{TEXT}`

Print TEXT in italic font. No effect in Info. See `Fonts`.

`\@ifclear FLAG`

If FLAG is cleared, the Texinfo formatting commands format text

between `'@ifclear FLAG'` and the following `'@end ifclear'` command. See `'@set'` `'@clear'` `'@value'`.

`'@ifinfo'`

Begin a stretch of text that will be ignored by TeX when it typesets the printed manual. The text appears only in the Info file. Pair with `'@end ifinfo'`. See Conditionally Visible Text.

`'@ifset FLAG'`

If FLAG is set, the Texinfo formatting commands format text between `'@ifset FLAG'` and the following `'@end ifset'` command. See `'@set'` `'@clear'` `'@value'`.

`'@iftex'`

Begin a stretch of text that will not appear in the Info file, but will be processed only by TeX. Pair with `'@end iftex'`. See Conditionally Visible Text.

`'@ignore'`

Begin a stretch of text that will not appear in either the Info file or the printed output. Pair with `'@end ignore'`. See Comments and Ignored Text.

`'@include FILENAME'`

Incorporate the contents of the file FILENAME into the Info file or printed document. See Include Files.

`'@inforef{NODE-NAME, [ENTRY-NAME], INFO-FILE-NAME}'`

Make a cross reference to an Info file for which there is no printed manual. See Cross references using `'@inforef'`.

`'\input MACRO-DEFINITIONS-FILE'`

Use the specified macro definitions file. This command is used only in the first line of a Texinfo file to cause TeX to make use of the `'texinfo'` macro definitions file. The backslash in `'\input'` is used instead of an `'@'` because TeX does not properly recognize `'@'` until after it has read the definitions file. See The Texinfo File Header.

`'@item'`

Indicate the beginning of a marked paragraph for `'@itemize'` and `'@enumerate'`; indicate the beginning of the text of a first column entry for `'@table'`, `'@ftable'`, and `'@vtable'`. See Lists and Tables.

`'@itemize MARK-GENERATING-CHARACTER-OR-COMMAND'`

Produce a sequence of indented paragraphs, with a mark inside the left margin at the beginning of each paragraph. Pair with `'@end itemize'`. See `'@itemize'`.

`'@itemx'`

Like `'@item'` but do not generate extra vertical space above the item text. See `'@itemx'`.

`'@kbd{KEYBOARD-CHARACTERS}'`

Indicate text that consists of characters of input to be typed by users. See `'@kbd'`.

`'@key{KEY-NAME}'`

Highlight KEY-NAME, a conventional name for a key on a keyboard.
See `'@key'`.

`'@kindex ENTRY'`

Add ENTRY to the index of keys. See
Defining the Entries of an Index.

`'@lisp'`

Begin an example of Lisp code. Indent text, do not fill, and
select fixed-width font. Pair with `'@end lisp'`. See `'@lisp'`.

`'@majorheading TITLE'`

Print a chapter-like heading in the text, but not in the table of
contents of a printed manual. Generate more vertical whitespace
before the heading than the `'@chapheading'` command. In Info, the
chapter heading line is underlined with asterisks. See
`'@majorheading'` and `'@chapheading'`.

`'@menu'`

Mark the beginning of a menu of nodes in Info. No effect in a
printed manual. Pair with `'@end menu'`. See Menus.

`'@minus{'`

Generate a minus sign. See `'@minus'`.

`'@need N'`

Start a new page in a printed manual if fewer than N mils
(thousandths of an inch) remain on the current page. See `'@need'`.

`'@node NAME, NEXT, PREVIOUS, UP'`

Define the beginning of a new node in Info, and serve as a locator
for references for TeX. See `'@node'`.

`'@noindent'`

Prevent text from being indented as if it were a new paragraph.
See `'@noindent'`.

`'@oddfooting [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page footings for odd-numbered (right-hand) pages. Not
relevant to Info. See How to Make Your Own Headings.

`'@oddheading [LEFT] @| [CENTER] @| [RIGHT]'`

Specify page headings for odd-numbered (right-hand) pages. Not
relevant to Info. See How to Make Your Own Headings.

`'@page'`

Start a new page in a printed manual. No effect in Info. See
`'@page'`.

`'@paragraphindent INDENT'`

Indent paragraphs by INDENT number of spaces; delete indentation
if the value of INDENT is 0; and do not change indentation if
INDENT is `'asis'`. See Paragraph Indenting.

`'@pindex ENTRY'`

Add ENTRY to the index of programs. See
Defining the Entries of an Index.

`'@point{}`

Indicate the position of point in a buffer to the reader with a
glyph: `'-!-'`. See Indicating Point in a Buffer.

`'@print{}`

Indicate printed output to the reader with a glyph: `'-|'`. See
Print Glyph.

`'@printindex INDEX-NAME'`

Print an alphabetized two-column index in a printed manual or
generate an alphabetized menu of index entries for Info. See
Printing Indices & Menus.

`'@pxref{NODE-NAME, [ENTRY], [TOPIC-OR-TITLE], [INFO-FILE], [MANUAL]}`

Make a reference that starts with a lower case 'see' in a printed
manual. Use within parentheses only. Do not follow command with a
punctuation mark. The Info formatting commands automatically
insert terminating punctuation as needed, which is why you do not
need to insert punctuation. Only the first argument is mandatory.
See `'@pxref'`.

`'@quotation'`

Narrow the margins to indicate text that is quoted from another
real or imaginary work. Write command on a line of its own. Pair
with `'@end quotation'`. See `'@quotation'`.

`'@r{TEXT}'`

Print TEXT in roman font. No effect in Info. See Fonts.

`'@ref{NODE-NAME, [ENTRY], [TOPIC-OR-TITLE], [INFO-FILE], [MANUAL]}`

Make a reference. In a printed manual, the reference does not
start with a 'See'. Follow command with a punctuation mark. Only
the first argument is mandatory. See `'@ref'`.

`'@refill'`

In Info, refill and indent the paragraph after all the other
processing has been done. No effect on TeX, which always refills.
This command is no longer needed, since all formatters now
automatically refill. See Refilling Paragraphs.

`'@result{}`

Indicate the result of an expression to the reader with a special
glyph: `'=>'`. See `'@result'`.

`'@samp{TEXT}'`

Highlight TEXT that is a literal example of a sequence of
characters. Used for single characters, for statements, and often
for entire shell commands. See `'@samp'`.

`'@sc{TEXT}'`

Set TEXT in a printed output in THE SMALL CAPS FONT and set text
in the Info file in uppercase letters. See Smallcaps.

`'@section TITLE'`

Begin a section within a chapter. In a printed manual, the section title is numbered and appears in the table of contents. In Info, the title is underlined with equal signs. See '@section'.

'@set FLAG [STRING]'

Make FLAG active, causing the Texinfo formatting commands to format text between subsequent pairs of '@ifset FLAG' and '@end ifset' commands. Optionally, set value of FLAG to STRING. See '@set' '@clear' '@value'.

'@setchapternewpage ON-OFF-ODD'

Specify whether chapters start on new pages, and if so, whether on odd-numbered (right-hand) new pages. See '@setchapternewpage'.

'@setfilename INFO-FILE-NAME'

Provide a name for the Info file. See General Syntactic Conventions.

'@settitle TITLE'

Provide a title for page headers in a printed manual. See General Syntactic Conventions.

'@shortcontents'

Print a short table of contents. Not relevant to Info, which uses menus rather than tables of contents. A synonym for '@summarycontents'. See Generating a Table of Contents.

'@smallbook'

Cause TeX to produce a printed manual in a 7 by 9.25 inch format rather than the regular 8.5 by 11 inch format. See Printing Small Books. Also, see '@smallexample' and '@smalllisp'.

'@smallexample'

Indent text to indicate an example. Do not fill, select fixed-width font. In '@smallbook' format, print text in a smaller font than with '@example'. Pair with '@end smallexample'. See '@smallexample' and '@smalllisp'.

'@smalllisp'

Begin an example of Lisp code. Indent text, do not fill, select fixed-width font. In '@smallbook' format, print text in a smaller font. Pair with '@end smalllisp'. See '@smallexample' and '@smalllisp'.

'@sp N'

Skip N blank lines. See '@sp'.

'@strong TEXT'

Emphasize TEXT by typesetting it in a **bold** font for the printed manual and by surrounding it with asterisks for Info. See Emphasizing Text.

'@subheading TITLE'

Print an unnumbered subsection-like heading in the text, but not in the table of contents of a printed manual. In Info, the title is underlined with hyphens. See

`'@unnumberedsubsec'` `'@appendixsubsec'` `'@subheading'`.

`'@subsection TITLE'`

Begin a subsection within a section. In a printed manual, the subsection title is numbered and appears in the table of contents. In Info, the title is underlined with hyphens. See `'@subsection'`.

`'@subsubheading TITLE'`

Print an unnumbered subsubsection-like heading in the text, but not in the table of contents of a printed manual. In Info, the title is underlined with periods. See The `'subsub'` Commands.

`'@subsubsection TITLE'`

Begin a subsubsection within a subsection. In a printed manual, the subsubsection title is numbered and appears in the table of contents. In Info, the title is underlined with periods. See The `'subsub'` Commands.

`'@subtitle TITLE'`

In a printed manual, set a subtitle in a normal sized font flush to the right-hand side of the page. Not relevant to Info, which does not have title pages. See `'@title'` `'@subtitle'` and `'@author'` Commands.

`'@summarycontents'`

Print a short table of contents. Not relevant to Info, which uses menus rather than tables of contents. A synonym for `'@shortcontents'`. See Generating a Table of Contents.

`'@syncodeindex FROM-INDEX INTO-INDEX'`

Merge the index named in the first argument into the index named in the second argument, printing the entries from the first index in `'@code'` font. See Combining Indices.

`'@synindex FROM-INDEX INTO-INDEX'`

Merge the index named in the first argument into the index named in the second argument. Do not change the font of FROM-INDEX entries. See Combining Indices.

`'@t{TEXT}'`

Print TEXT in a fixed-width, typewriter-like font. No effect in Info. See Fonts.

`'@table FORMATTING-COMMAND'`

Begin a two-column table, using `'@item'` for each entry. Write each first column entry on the same line as `'@item'`. First column entries are printed in the font resulting from FORMATTING-COMMAND. Pair with `'@end table'`. See Making a Two-column Table. Also see See `'@ftable'` and `'@vtable'`, and See `'@itemx'`.

`'@TeX{}'`

Insert the logo TeX. See Inserting TeX and (C).

`'@tex'`

Enter TeX completely. Pair with `'@end tex'`. See Using Ordinary TeX Commands.

`\@thischapter'`

In a heading or footing, stands for the number and name of the current chapter, in the format `'Chapter 1: Title'`. See `How to Make Your Own Headings`.

`\@thischaptername'`

In a heading or footing, stands for the name of the current chapter. See `How to Make Your Own Headings`.

`\@thisfile'`

In a heading or footing, stands for the name of the current `'@include'` file. Does not insert anything if not within an `'@include'` file. See `How to Make Your Own Headings`.

`\@thispage'`

In a heading or footing, stands for the current page number. See `How to Make Your Own Headings`.

`\@thistitle'`

In a heading or footing, stands for the name of the document, as specified by the `'@settitle'` command. See `How to Make Your Own Headings`.

`\@index ENTRY'`

Add ENTRY to the index of data types. See `Defining the Entries of an Index`.

`\@title TITLE'`

In a printed manual, set a title flush to the left-hand side of the page in a larger than normal font and underline it with a black rule. Not relevant to Info, which does not have title pages. See `The '@title' '@subtitle' and '@author' Commands`.

`\@titlefont{TEXT}'`

In a printed manual, print TEXT in a larger than normal font. Not relevant to Info, which does not have title pages. See `The '@titlefont' '@center' and '@sp' Commands`.

`\@titlepage'`

Indicate to Texinfo the beginning of the title page. Write command on a line of its own. Pair with `'@end titlepage'`. Nothing between `'@titlepage'` and `'@end titlepage'` appears in Info. See `'@titlepage'`.

`\@today{'`

Insert the current date, in `'1 Jan 1900'` style. See `How to Make Your Own Headings`.

`\@top TITLE'`

In a Texinfo file to be formatted with `'makeinfo'`, identify the topmost `'@node'` line in the file, which must be written on the line immediately preceding the `'@top'` command. Used for `'makeinfo''s` node pointer insertion feature. The title is underlined with asterisks. Both the `'@node'` line and the `'@top'` line normally should be enclosed by `'@ifinfo'` and `'@end ifinfo'`. In TeX and `'texinfo-format-buffer'`, the `'@top'` command is merely a synonym for `'@unnumbered'`. See `Creating Pointers with 'makeinfo'`.

`'@unnumbered TITLE'`

In a printed manual, begin a chapter that appears without chapter numbers of any kind. The title appears in the table of contents of a printed manual. In Info, the title is underlined with asterisks. See `'@unnumbered'` and `'@appendix'`.

`'@unnumberedsec TITLE'`

In a printed manual, begin a section that appears without section numbers of any kind. The title appears in the table of contents of a printed manual. In Info, the title is underlined with equal signs. See Section Commands.

`'@unnumberedsubsec TITLE'`

In a printed manual, begin an unnumbered subsection within a chapter. The title appears in the table of contents of a printed manual. In Info, the title is underlined with hyphens. See `'@unnumberedsubsec'` `'@appendixsubsec'` `'@subheading'`.

`'@unnumberedsubsubsec TITLE'`

In a printed manual, begin an unnumbered subsubsection within a chapter. The title appears in the table of contents of a printed manual. In Info, the title is underlined with periods. See The `'subsub'` Commands.

`'@value{FLAG}'`

Replace FLAG with the value to which it is set by `'@set FLAG'`. See `'@set'` `'@clear'` `'@value'`.

`'@var{METASYNTACTIC-VARIABLE}'`

Highlight a metasyntactic variable, which is something that stands for another piece of text. See Indicating Metasyntactic Variables.

`'@vindex ENTRY'`

Add ENTRY to the index of variables. See Defining the Entries of an Index.

`'@vskip AMOUNT'`

In a printed manual, insert whitespace so as to push text on the remainder of the page towards the bottom of the page. Used in formatting the copyright page with the argument `'0pt plus 1filll'`. (Note spelling of `'filll'`.) `'@vskip'` may be used only in contexts ignored for Info. See The Copyright Page and Printed Permissions.

`'@vtable FORMATTING-COMMAND'`

Begin a two-column table, using `'@item'` for each entry. Automatically enter each of the items in the first column into the index of variables. Pair with `'@end vtable'`. The same as `'@table'`, except for indexing. See `'@ftable'` and `'@vtable'`.

`'@w{TEXT}'`

Prevent TEXT from being split across two lines. Do not end a paragraph that uses `'@w'` with an `'@refill'` command. In the Texinfo file, keep TEXT on one line. See `'@w'`.

`'@xref{NODE-NAME, [ENTRY], [TOPIC-OR-TITLE], [INFO-FILE], [MANUAL]}'`

Make a reference that starts with 'See' in a printed manual. Follow command with a punctuation mark. Only the first argument is mandatory. See '@xref'.

1.217 texi.guide/Tips

Tips and Hints

Here are some tips for writing Texinfo documentation:

- * Write in the present tense, not in the past or the future.
- * Write actively! For example, write "We recommend that ..." rather than "It is recommended that ...".
- * Use 70 or 72 as your fill column. Longer lines are hard to read.
- * Include a copyright notice and copying permissions.

Index, index, index!

.....

Write many index entries, in different ways. Readers like indices; they are helpful and convenient.

Although it is easiest to write index entries as you write the body of the text, some people prefer to write entries afterwards. In either case, write an entry before the paragraph to which it applies. This way, an index entry points to the first page of a paragraph that is split across pages.

Here are more hints we have found valuable:

- * Write each index entry differently, so each entry refers to a different place in the document. The index of an Info file lists only one location for each entry.
 - * Write index entries only where a topic is discussed significantly. For example, it is not useful to index "debugging information" in a chapter on reporting bugs. Someone who wants to know about debugging information will certainly not find it in that chapter.
 - * Consistently capitalize the first word of every index entry, or else use lower case. According to convention, you should capitalize the first word of an index entry. However, this practice may make an index look crowded. Some writers prefer lower case. Regardless of which you prefer, choose one style and stick to it. Mixing the two styles looks bad.
 - * Always capitalize or use upper case for those words in an index for which this is proper, such as names of countries or acronyms.
 - * Write the indexing commands that refer to a whole section
-

immediately after the section command, and write the indexing commands that refer to the paragraph before the paragraph.

In the example that follows, a blank line comes after the index entry for "Leaping":

```
@section The Dog and the Fox
@cindex Jumping, in general
@cindex Leaping

@cindex Dog, lazy, jumped over
@cindex Lazy dog jumped over
@cindex Fox, jumps over dog
@cindex Quick fox jumps over dog
The quick brown fox jumps over the lazy dog.
```

(Note that the example shows entries for the same concept that are written in different ways--'Lazy dog', and 'Dog, lazy'--so readers can look up the concept in different ways.)

Blank lines

.....

- * Insert a blank line between a sectioning command and the first following sentence or paragraph, or between the indexing commands associated with the sectioning command and the first following sentence or paragraph, as shown in the tip on indexing. Otherwise, a formatter may fold title and paragraph together.
- * Always insert a blank line before an '@table' command and after an '@end table' command; but never insert a blank line after an '@table' command or before an '@end table' command.

For example,

```
Types of fox:

@table @samp
@item Quick
Jump over lazy dogs.

@item Brown
Also jump over lazy dogs.
@end table
@noindent
On the other hand, ...
```

Insert blank lines before and after '@itemize' ... '@end itemize' and '@enumerate' ... '@end enumerate' in the same way.

Complete phrases

.....

Complete phrases are easier to read than ...

- * Write entries in an itemized list as complete sentences; or at least, as complete phrases. Incomplete expressions ... awkward

... like this.

- * Write the prefatory sentence or phrase for a multi-item list or table as a complete expression. Do not write "You can set:"; instead, write "You can set these variables:". The former expression sounds cut off.

Editions, dates and versions

.....

Write the edition and version numbers and date in three places in every manual:

1. In the first '@ifinfo' section, for people reading the Texinfo file.
2. In the '@titlepage' section, for people reading the printed manual.
3. In the 'Top' node, for people reading the Info file.

Also, it helps to write a note before the first '@ifinfo' section to explain what you are doing.

For example:

```
@c ==> NOTE! <==
@c Specify the edition and version numbers and date
@c in *three* places:
@c 1. First ifinfo section 2. title page 3. top node
@c To find the locations, search for !!set

@ifinfo
@c !!set edition, date, version
This is Edition 4.03, January 1992,
of the @cite{GDB Manual} for GDB Version 4.3.
...
```

--or use '@set' and '@value' (see '@value' Example).

Definition Commands

.....

Definition commands are '@defn', '@defun', '@defmac', and the like, and enable you to write descriptions in a uniform format.

- * Write just one definition command for each entity you define with a definition command. The automatic indexing feature creates an index entry that leads the reader to the definition.
- * Use '@table' ... '@end table' in an appendix that contains a summary of functions, not '@defn' or other definition commands.

Capitalization

.....

- * Capitalize 'Texinfo'; it is a name. Do not write the 'x' or 'i' in upper case.

* Capitalize `'Info'`; it is a name.

* Write TeX using the `'@TeX{...}'` command. Note the uppercase `'T'` and `'X'`. This command causes the formatters to typeset the name according to the wishes of Donald Knuth, who wrote TeX.

Spaces

.....

Do not use spaces to format a Texinfo file, except inside of `'@example'` ... `'@end example'` and similar commands.

For example, TeX fills the following:

```
@kbd{C-x v}
@kbd{M-x vc-next-action}
  Perform the next logical operation
  on the version-controlled file
  corresponding to the current buffer.
```

so it looks like this:

```
'C-x v' 'M-x vc-next-action' Perform the next logical operation on
the version-controlled file corresponding to the current buffer.
```

In this case, the text should be formatted with `'@table'`, `'@item'`, and `'@itemx'`, to create a table.

.....

* Use `'@code'` around Lisp symbols, including command names. For example,

```
The main function is @code{vc-next-action}, ...
```

* Avoid putting letters such as `'s'` immediately after an `'@code'`. Such letters look bad.

* Use `'@var'` around meta-variables. Do not write angle brackets around them.

* Use three hyphens in a row, `'---'`, to indicate a long dash. TeX typesets these as a long dash and the Info formatters reduce three hyphens to two.

Periods Outside of Quotes

.....

Place periods and other punctuation marks *outside* of quotations, unless the punctuation is part of the quotation. This practice goes against convention, but enables the reader to distinguish between the contents of the quotation and the whole passage.

For example, you should write the following sentence with the period outside the end quotation marks:

Evidently, `'au'` is an abbreviation for `''author''`.

since `'au'` does *not* serve as an abbreviation for `'author.'` (with a period following the word).

Introducing New Terms

.....

- * Introduce new terms so that a user who does not know them can understand them from context; or write a definition for the term.

For example, in the following, the terms "check in", "register" and "delta" are all appearing for the first time; the example sentence should be rewritten so they are understandable.

The major function assists you in checking in a file to your version control system and registering successive sets of changes to it as deltas.

- * Use the `'@dfn'` command around a word being introduced, to indicate that the user should not expect to know the meaning already, and should expect to learn the meaning from this passage.

.....

Absolutely never use `'@pxref'` except in the special context for which it is designed: inside parentheses, with the closing parenthesis following immediately after the closing brace. One formatter automatically inserts closing punctuation and the other does not. This means that the output looks right both in printed output and in an Info file, but only when the command is used inside parentheses.

Invoking from a Shell

.....

You can invoke programs such as Emacs, GCC, and GAWK from a shell. The documentation for each program should contain a section that describes this. Unfortunately, if the node names and titles for these sections are all different, readers find it hard to search for the section.

Name such sections with a phrase beginning with the word `'Invoking ...'`, as in `'Invoking Emacs'`; this way users can find the section easily.

ANSI C Syntax

.....

When you use `'@example'` to describe a C function's calling conventions, use the ANSI C syntax, like this:

```
void dld_init (char *@var{path});
```

And in the subsequent discussion, refer to the argument values by writing the same argument names, again highlighted with `'@var'`.

Avoid the obsolete style that looks like this:

```
#include <dld.h>
```

```
dld_init (path)
char *path;
```

Also, it is best to avoid writing `'#include'` above the declaration just to indicate that the function is declared in a header file. The practice may give the misimpression that the `'#include'` belongs near the declaration of the function. Either state explicitly which header file holds the declaration or, better yet, name the header file used for a group of functions at the beginning of the section that describes the functions.

Bad Examples

.....

Here are several examples of bad writing to avoid:

In this example, say, " ... you must `'@dfn'`{check in} the new version." That flows better.

When you are done editing the file, you must perform a `'@dfn'`{check in}.

In the following example, say, "... makes a unified interface such as VC mode possible."

SCCS, RCS and other version-control systems all perform similar functions in broadly similar ways (it is this resemblance which makes a unified control mode like this possible).

And in this example, you should specify what `'it'` refers to:

If you are working with other people, it assists in coordinating everyone's changes so they do not step on each other.

And Finally ...

.....

- * Pronounce TeX as if the `'X'` were a Greek `'chi'`, as the last sound in the name `'Bach'`. But pronounce Texinfo as in `'speck'`: `'teckinfo'`.

- * Write notes for yourself at the very end of a Texinfo file after the `'@bye'`. None of the formatters process text after the `'@bye'`; it is as if the text were within `'@ignore'` ... `'@end ignore'`.

1.218 texi.guide/Sample Texinfo File

A Sample Texinfo File

Here is a complete, short sample Texinfo file, without any

commentary. You can see this file, with comments, in the first chapter. See A Short Sample Texinfo File.

```
\input texinfo    @c -*-texinfo*-
@c %**start of header
@setfilename sample.info
@settitle Sample Document
@c %**end of header

@setchapternewpage odd

@ifinfo
This is a short example of a complete Texinfo file.

Copyright 1990 Free Software Foundation, Inc.
@end ifinfo

@titlepage
@sp 10
@comment The title is printed in a large font.
@center @titlefont{Sample Title}

@c The following two commands start the copyright page.
@page
@vskip 0pt plus 1filll
Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end titlepage

@node    Top,          First Chapter, (dir),    (dir)
@comment node-name, next,          previous, up

@menu
* First Chapter::      The first chapter is the
                        only chapter in this sample.
* Concept Index::      This index has two entries.
@end menu

@node    First Chapter, Concept Index, Top,      Top
@comment node-name,      next,          previous, up
@chapter First Chapter
@cindex Sample index entry

This is the contents of the first chapter.
@cindex Another sample index entry

Here is a numbered list.

@enumerate
@item
This is the first item.

@item
This is the second item.
@end enumerate

The @code{makeinfo} and @code{texinfo-format-buffer}
commands transform a Texinfo file such as this into
```

an Info file; and @TeX{} typesets it for a printed manual.

```
@node      Concept Index,      , First Chapter, Top
@comment node-name,      next, previous,      up
@unnumbered Concept Index
```

```
@printindex cp
```

```
@contents
```

```
@bye
```

1.219 texi.guide/Sample Permissions

Sample Permissions

Texinfo files should contain sections that tell the readers that they have the right to copy and distribute the Texinfo file, the Info file, and the printed manual.

Also, if you are writing a manual about software, you should explain that the software is free and either include the GNU General Public License (GPL) or provide a reference to it. See Distribution, for an example of the text that could be used in the software "Distribution", "General Public License", and "NO WARRANTY" sections of a document. See Texinfo Copying Conditions, for an example of a brief explanation of how the copying conditions provide you with rights.

Inserting Permissions	How to put permissions in your document.
ifinfo Permissions	Sample 'ifinfo' copying permissions.
Titlepage Permissions	Sample Titlepage copying permissions.

1.220 texi.guide/Inserting Permissions

Inserting Permissions

=====

In a Texinfo file, the first '@ifinfo' section usually begins with a line that says what the file documents. This is what a person reading the unprocessed Texinfo file or using the advanced Info command 'g *' sees first. See Advanced Info commands, for more information. (A reader using the regular Info commands usually starts reading at the first node and skips this first section, which is not in a node.)

In the '@ifinfo' section, the summary sentence is followed by a copyright notice and then by the copying permission notice. One of the copying permission paragraphs is enclosed in '@ignore' and '@end ignore' commands. This paragraph states that the Texinfo file can be

processed through TeX and printed, provided the printed manual carries the proper copying permission notice. This paragraph is not made part of the Info file since it is not relevant to the Info file; but it is a mandatory part of the Texinfo file since it permits people to process the Texinfo file in TeX and print the results.

In the printed manual, the Free Software Foundation copying permission notice follows the copyright notice and publishing information and is located within the region delineated by the '@titlepage' and '@end titlepage' commands. The copying permission notice is exactly the same as the notice in the '@ifinfo' section except that the paragraph enclosed in '@ignore' and '@end ignore' commands is not part of the notice.

To make it simple to insert a permission notice into each section of the Texinfo file, sample permission notices for each section are reproduced in full below.

Note that you may need to specify the correct name of a section mentioned in the permission notice. For example, in 'The GDB Manual', the name of the section referring to the General Public License is called the "GDB General Public License", but in the sample shown below, that section is referred to generically as the "GNU General Public License". If the Texinfo file does not carry a copy of the General Public License, leave out the reference to it, but be sure to include the rest of the sentence.

1.221 texi.guide/ifinfo Permissions

'ifinfo' Copying Permissions
=====

In the '@ifinfo' section of a Texinfo file, the standard Free Software Foundation permission notice reads as follows:

This file documents ...

Copyright 1992 Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

@ignore

Permission is granted to process this file through TeX and print the results, provided the printed document carries a copying permission notice identical to this one except for the removal of this paragraph (this paragraph not being relevant to the printed manual).

@end ignore

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the sections

entitled ``Copying'' and ``GNU General Public License'' are included exactly as in the original, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Free Software Foundation.

1.222 texi.guide/Titlepage Permissions

Titlepage Copying Permissions
=====

In the '@titlepage' section of a Texinfo file, the standard Free Software Foundation copying permission notice follows the copyright notice and publishing information. The standard phrasing is as follows:

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the sections entitled ``Copying'' and ``GNU General Public License'' are included exactly as in the original, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Free Software Foundation.

1.223 texi.guide/Include Files

Include Files

When TeX or an Info formatting command sees an '@include' command in a Texinfo file, it processes the contents of the file named by the command and incorporates them into the DVI or Info file being created. Index entries from the included file are incorporated into the indices of the output file.

Include files let you keep a single large document as a collection of conveniently small parts.

Using Include Files	How to use the '@include' command.
texinfo-multiple-files-update	How to create and update nodes and menus when using included files.
Include File Requirements	What 'texinfo-multiple-files-update' expects.
Sample Include File	A sample outer file with included files within it; and a sample included file.
Include Files Evolution	How use of the '@include' command has changed over time.

1.224 texi.guide/Using Include Files

How to Use Include Files

=====

To include another file within a Texinfo file, write the '@include' command at the beginning of a line and follow it on the same line by the name of a file to be included. For example:

```
@include buffers.texi
```

An included file should simply be a segment of text that you expect to be included as is into the overall or "outer" Texinfo file; it should not contain the standard beginning and end parts of a Texinfo file. In particular, you should not start an included file with a line saying '\input texinfo'; if you do, that phrase is inserted into the output file as is. Likewise, you should not end an included file with an '@bye' command; nothing after '@bye' is formatted.

In the past, you were required to write an '@setfilename' line at the beginning of an included file, but no longer. Now, it does not matter whether you write such a line. If an '@setfilename' line exists in an included file, it is ignored.

Conventionally, an included file begins with an '@node' line that is followed by an '@chapter' line. Each included file is one chapter. This makes it easy to use the regular node and menu creating and updating commands to create the node pointers and menus within the included file. However, the simple Emacs node and menu creating and updating commands do not work with multiple Texinfo files. Thus you cannot use these commands to fill in the 'Next', 'Previous', and 'Up' pointers of the '@node' line that begins the included file. Also, you cannot use the regular commands to create a master menu for the whole file. Either you must insert the menus and the 'Next', 'Previous', and 'Up' pointers by hand, or you must use the GNU Emacs Texinfo mode command, 'texinfo-multiple-files-update', that is designed for '@include' files.

1.225 texi.guide/texinfo-multiple-files-update

`'texinfo-multiple-files-update'`
=====

GNU Emacs Texinfo mode provides a command to handle included files called `'texinfo-multiple-files-update'`. This command creates or updates `'Next'`, `'Previous'`, and `'Up'` pointers of included files as well as those in the outer or overall Texinfo file, and it creates or updates a main menu in the outer file. Depending whether you call it with optional arguments, the command updates only the pointers in the first `'@node'` line of the included files or all of them:

`'M-x texinfo-multiple-files-update'`
Called without any arguments:

- Create or update the `'Next'`, `'Previous'`, and `'Up'` pointers of the first `'@node'` line in each file included in an outer or overall Texinfo file.
- Create or update the `'Top'` level node pointers of the outer or overall file.
- Create or update a main menu in the outer file.

`'C-u M-x texinfo-multiple-files-update'`
Called with `'C-u'` as a prefix argument:

- Create or update pointers in the first `'@node'` line in each included file.
- Create or update the `'Top'` level node pointers of the outer file.
- Create and insert a master menu in the outer file. The master menu is made from all the menus in all the included files.

`'C-u 8 M-x texinfo-multiple-files-update'`
Called with a numeric prefix argument, such as `'C-u 8'`:

- Create or update **all** the `'Next'`, `'Previous'`, and `'Up'` pointers of all the included files.
- Create or update **all** the menus of all the included files.
- Create or update the `'Top'` level node pointers of the outer or overall file.
- And then create a master menu in the outer file. This is similar to invoking `'texinfo-master-menu'` with an argument when you are working with just one file.

Note the use of the prefix argument in interactive use: with a regular prefix argument, just `'C-u'`, the `'texinfo-multiple-files-update'` command inserts a master menu; with a

numeric prefix argument, such as ``C-u 8'`, the command updates **every** pointer and menu in **all** the files and then inserts a master menu.

1.226 texi.guide/Include File Requirements

Include File Requirements

=====

If you plan to use the ``texinfo-multiple-files-update'` command, the outer Texinfo file that lists included files within it should contain nothing but the beginning and end parts of a Texinfo file, and a number of ``@include'` commands listing the included files. It should not even include indices, which should be listed in an included file of their own.

Moreover, each of the included files must contain exactly one highest level node (conventionally, ``@chapter'` or equivalent), and this node must be the first node in the included file. Furthermore, each of these highest level nodes in each included file must be at the same hierarchical level in the file structure. Usually, each is an ``@chapter'`, an ``@appendix'`, or an ``@unnumbered'` node. Thus, normally, each included file contains one, and only one, chapter or equivalent-level node.

The outer file should contain only **one** node, the ``Top'` node. It should **not** contain any nodes besides the single ``Top'` node. The ``texinfo-multiple-files-update'` command will not process them.

1.227 texi.guide/Sample Include File

Sample File with ``@include'`

=====

Here is an example of a complete outer Texinfo file with ``@include'` files within it before running ``texinfo-multiple-files-update'`, which would insert a main or master menu:

```
\input texinfo @c -*-texinfo*-
@setfilename include-example.info
@settitle Include Example

@setchapternewpage odd
@titlepage
@sp 12
@center @titlefont{Include Example}
@sp 2
@center by Whom Ever

@page
@vskip 0pt plus 1filll
```

```
Copyright @copyright{} 1990 Free Software Foundation, Inc.
@end titlepage
```

```
@ifinfo
@node Top, First, (dir), (dir)
@top Master Menu
@end ifinfo
```

```
@include foo.texinfo
@include bar.texinfo
@include concept-index.texinfo
```

```
@summarycontents
@contents
```

```
@bye
```

An included file, such as 'foo.texinfo', might look like this:

```
@node First, Second, , Top
@chapter First Chapter
```

```
Contents of first chapter ...
```

The full contents of 'concept-index.texinfo' might be as simple as this:

```
@node Concept Index, , Second, Top
@unnumbered Concept Index
```

```
@printindex cp
```

The outer Texinfo source file for 'The GNU Emacs Lisp Reference Manual' is named 'elisp.texi'. This outer file contains a master menu with 417 entries and a list of 41 '@include' files.

1.228 texi.guide/Include Files Evolution

Evolution of Include Files

=====

When Info was first created, it was customary to create many small Info files on one subject. Each Info file was formatted from its own Texinfo source file. This custom meant that Emacs did not need to make a large buffer to hold the whole of a large Info file when someone wanted information; instead, Emacs allocated just enough memory for the small Info file that contained the particular information sought. This way, Emacs could avoid wasting memory.

References from one file to another were made by referring to the file name as well as the node name. (See Referring to Other Info Files. Also, see See '@xref' with Four and Five Arguments.)

Include files were designed primarily as a way to create a single,

large printed manual out of several smaller Info files. In a printed manual, all the references were within the same document, so TeX could automatically determine the references' page numbers. The Info formatting commands used include files only for creating joint indices; each of the individual Texinfo files had to be formatted for Info individually. (Each, therefore, required its own '@setfilename' line.)

However, because large Info files are now split automatically, it is no longer necessary to keep them small.

Nowadays, multiple Texinfo files are used mostly for large documents, such as 'The GNU Emacs Lisp Reference Manual', and for projects in which several different people write different sections of a document simultaneously.

In addition, the Info formatting commands have been extended to work with the '@include' command so as to create a single large Info file that is split into smaller files if necessary. This means that you can write menus and cross references without naming the different Texinfo files.

1.229 texi.guide/Headings

Page Headings

Most printed manuals contain headings along the top of every page except the title and copyright pages. Some manuals also contain footings. (Headings and footings have no meaning to Info, which is not paginated.)

Headings Introduced	Conventions for using page headings.
Heading Format	Standard page heading formats.
Heading Choice	How to specify the type of page heading.
Custom Headings	How to create your own headings and footings.

1.230 texi.guide/Headings Introduced

Headings Introduced

=====

Texinfo provides standard page heading formats for manuals that are printed on one side of each sheet of paper and for manuals that are printed on both sides of the paper. Usually, you will use one or other of these formats, but you can specify your own format, if you wish.

In addition, you can specify whether chapters should begin on a new page, or merely continue the same page as the previous chapter; and if chapters begin on new pages, you can specify whether they must be

odd-numbered pages.

By convention, a book is printed on both sides of each sheet of paper. When you open a book, the right-hand page is odd-numbered, and chapters begin on right-hand pages--a preceding left-hand page is left blank if necessary. Reports, however, are often printed on just one side of paper, and chapters begin on a fresh page immediately following the end of the preceding chapter. In short or informal reports, chapters often do not begin on a new page at all, but are separated from the preceding text by a small amount of whitespace.

The '@setchapternewpage' command controls whether chapters begin on new pages, and whether one of the standard heading formats is used. In addition, Texinfo has several heading and footing commands that you can use to generate your own heading and footing formats.

In Texinfo, headings and footings are single lines at the tops and bottoms of pages; you cannot create multiline headings or footings. Each header or footer line is divided into three parts: a left part, a middle part, and a right part. Any part, or a whole line, may be left blank. Text for the left part of a header or footer line is set flushleft; text for the middle part is centered; and, text for the right part is set flushright.

1.231 texi.guide/Heading Format

Standard Heading Formats
=====

Texinfo provides two standard heading formats, one for manuals printed on one side of each sheet of paper, and the other for manuals printed on both sides of the paper.

By default, nothing is specified for the footing of a Texinfo file, so the footing remains blank.

The standard format for single-sided printing consists of a header line in which the left-hand part contains the name of the chapter, the central part is blank, and the right-hand part contains the page number.

A single-sided page looks like this:

```
|-----|
| chapter    page number |
|-----|
| Start of text ...      |
| ...                    |
|-----|
```

The standard format for two-sided printing depends on whether the page number is even or odd. By convention, even-numbered pages are on the left- and odd-numbered pages are on the right. (TeX will adjust the widths of the left- and right-hand margins. Usually, widths are

correct, but during double-sided printing, it is wise to check that pages will bind properly--sometimes a printer will produce output in which the even-numbered pages have a larger right-hand margin than the odd-numbered pages.)

In the standard double-sided format, the left part of the left-hand (even-numbered) page contains the page number, the central part is blank, and the right part contains the title (specified by the '@settitle' command). The left part of the right-hand (odd-numbered) page contains the name of the chapter, the central part is blank, and the right part contains the page number.

Two pages, side by side as in an open book, look like this:

	page number	title	
	Start of text ...		
	...		

	chapter	page number	
	More text ...		
	...		

The chapter name is preceded by the word 'Chapter', the chapter number and a colon. This makes it easier to keep track of where you are in the manual.

1.232 texi.guide/Heading Choice

Specifying the Type of Heading

=====

TeX does not begin to generate page headings for a standard Texinfo file until it reaches the '@end titlepage' command. Thus, the title and copyright pages are not numbered. The '@end titlepage' command causes TeX to begin to generate page headings according to a standard format specified by the '@setchapternewpage' command that precedes the '@titlepage' section.

There are four possibilities:

No '@setchapternewpage' command

Cause TeX to specify the single-sided heading format, with chapters on new pages. This is the same as '@setchapternewpage on'.

'@setchapternewpage on'

Specify the single-sided heading format, with chapters on new pages.

'@setchapternewpage off'

Cause TeX to start a new chapter on the same page as the last page of the preceding chapter, after skipping some vertical whitespace. Also cause TeX to typeset for single-sided printing. (You can override the headers format with the '@headings double' command; see See The '@headings' Command.)

```
'@setchapternewpage odd'
```

Specify the double-sided heading format, with chapters on new pages.

Texinfo lacks an '@setchapternewpage even' command.

1.233 texi.guide/Custom Headings

How to Make Your Own Headings

=====

You can use the standard headings provided with Texinfo or specify your own.

Texinfo provides six commands for specifying headings and footings. The '@everyheading' command and '@everyfooting' command generate page headers and footers that are the same for both even- and odd-numbered pages. The '@evenheading' command and '@evenfooting' command generate headers and footers for even-numbered (left-hand) pages; and the '@oddheading' command and '@oddfooting' command generate headers and footers for odd-numbered (right-hand) pages.

Write custom heading specifications in the Texinfo file immediately after the '@end titlepage' command. Enclose your specifications between '@iftex' and '@end iftex' commands since the 'texinfo-format-buffer' command may not recognize them. Also, you must cancel the predefined heading commands with the '@headings off' command before defining your own specifications.

Here is how to tell TeX to place the chapter name at the left, the page number in the center, and the date at the right of every header for both even- and odd-numbered pages:

```
@iftex
@headings off
@everyheading @thischapter @| @thispage @| @today{}
@end iftex
```

You need to divide the left part from the central part and the central part from the right had part by inserting '@|' between parts. Otherwise, the specification command will not be able to tell where the text for one part ends and the next part begins.

Each part can contain text or @-commands. The text is printed as if the part were within an ordinary paragraph in the body of the page. The @-commands replace themselves with the page number, date, chapter name, or whatever.

Here are the six heading and footing commands:

```
'@everyheading LEFT @| CENTER @| RIGHT'
 '@everyfooting LEFT @| CENTER @| RIGHT'
```

The 'every' commands specify the format for both even- and

odd-numbered pages. These commands are for documents that are printed on one side of each sheet of paper, or for documents in which you want symmetrical headers or footers.

```
'@evenheading LEFT @| CENTER @| RIGHT'
'@oddheading  LEFT @| CENTER @| RIGHT'
'@evenfooting LEFT @| CENTER @| RIGHT'
'@oddfooting  LEFT @| CENTER @| RIGHT'
```

The 'even' and 'odd' commands specify the format for even-numbered pages and odd-numbered pages. These commands are for books and manuals that are printed on both sides of each sheet of paper.

Use the '@this...' series of @-commands to provide the names of chapters and sections and the page number. You can use the '@this...' commands in the left, center, or right portions of headers and footers, or anywhere else in a Texinfo file so long as they are between '@iftex' and '@end iftex' commands.

Here are the '@this...' commands:

```
'@thispage'
  Expands to the current page number.

'@thischaptername'
  Expands to the name of the current chapter.

'@thischapter'
  Expands to the number and name of the current chapter, in the
  format 'Chapter 1: Title'.

'@thistitle'
  Expands to the name of the document, as specified by the
  '@settitle' command.

'@thisfile'
  For '@include' files only: expands to the name of the current
  '@include' file. If the current Texinfo source file is not an
  '@include' file, this command has no effect. This command does
  *not* provide the name of the current Texinfo source file unless
  it is an '@include' file. (See Include Files, for more
  information about '@include' files.)
```

You can also use the '@today{}' command, which expands to the current date, in '1 Jan 1900' format.

Other @-commands and text are printed in a header or footer just as if they were in the body of a page. It is useful to incorporate text, particularly when you are writing drafts:

```
@iftex
@headings off
@everyheading @emph{Draft!} @| @thispage @| @thischapter
@everyfooting @| @| Version: 0.27: @today{}
@end iftex
```

Beware of overlong titles: they may overlap another part of the header or footer and blot it out.

1.234 texi.guide/Catching Mistakes

Formatting Mistakes

Besides mistakes in the content of your documentation, there are two kinds of mistake you can make with Texinfo: you can make mistakes with and chapters.

Emacs has two tools for catching the @-command mistakes and two for catching structuring mistakes.

For finding problems with @-commands, you can run TeX or a region formatting command on the region that has a problem; indeed, you can run these commands on each region as you write it.

For finding problems with the structure of nodes and chapters, you can use 'C-c C-s' ('texinfo-show-structure') and the related 'occur' command and you can use the 'M-x Info-validate' command.

makeinfo preferred	'makeinfo' finds errors.
Debugging with Info	How to catch errors with Info formatting.
Debugging with TeX	How to catch errors with TeX formatting.
Using texinfo-show-structure	How to use 'texinfo-show-structure'.
Using occur	How to list all lines containing a pattern.
Running Info-Validate	How to find badly referenced nodes.

1.235 texi.guide/makeinfo preferred

'makeinfo' Find Errors

=====

The 'makeinfo' program does an excellent job of catching errors and reporting them--far better than either the 'texinfo-format-region' or the 'texinfo-format-buffer' command. In addition, the various functions for automatically creating and updating node pointers and menus remove many opportunities for human error.

If you can, use the updating commands to create and insert pointers and menus. These prevent many errors. Then use 'makeinfo' (or its Texinfo mode manifestations, 'makeinfo-region' and 'makeinfo-buffer') to format your file and check for other errors. This is the best way to work with Texinfo. But if you cannot use 'makeinfo', or your problem is very puzzling, then you may want to use the tools described in this appendix.

1.236 texi.guide/Debugging with Info

Catching Errors with Info Formatting

After you have written part of a Texinfo file, you can use the `'texinfo-format-region'` or the `'makeinfo-region'` command to see whether the region formats properly.

Most likely, however, you are reading this section because for some reason you cannot use the `'makeinfo-region'` command; therefore, the rest of this section presumes that you are using `'texinfo-format-region'`.

If you make a mistake with an `@`-command, `'texinfo-format-region'` will stop processing at or after the error and display an error message. To see where in the buffer the error occurred, switch to the `'*Info Region*' buffer; the cursor will be in a position that is after the location of the error. Also, the text will not be formatted after the place where the error occurred (or more precisely, where it was detected).`

For example, if you accidentally end a menu with the command `'@end menus'` with an `'s'` on the end, instead of with `'@end menu'`, you will see an error message that says:

```
@end menus is not handled by texinfo
```

The cursor will stop at the point in the buffer where the error occurs, or not long after it. The buffer will look like this:

```
----- Buffer: *Info Region* -----
* Menu:

* Using texinfo-show-structure::  How to use
                                'texinfo-show-structure'
                                to catch mistakes.
* Running Info-Validate::        How to check for
                                unreferenced nodes.

@end menus
-!-
----- Buffer: *Info Region* -----
```

The `'texinfo-format-region'` command sometimes provides slightly odd error messages. For example, the following cross reference fails to format:

```
(@xref{Catching Mistakes, for more info.})
```

In this case, `'texinfo-format-region'` detects the missing closing brace but displays a message that says `'Unbalanced parentheses'` rather than `'Unbalanced braces'`. This is because the formatting command looks for mismatches between braces as if they were parentheses.

Sometimes `'texinfo-format-region'` fails to detect mistakes. For example, in the following, the closing brace is swapped with the

closing parenthesis:

```
(@xref{Catching Mistakes), for more info.)
```

Formatting produces:

```
(*Note for more info.: Catching Mistakes)
```

The only way for you to detect this error is to realize that the reference should have looked like this:

```
(*Note Catching Mistakes::, for more info.)
```

Incidentally, if you are reading this node in Info and type 'f RET' ('Info-follow-reference'), you will generate an error message that says:

```
No such node: "Catching Mistakes) The only way ...
```

This is because Info perceives the example of the error as the first cross reference in this node and if you type a RET immediately after typing the Info 'f' command, Info will attempt to go to the referenced node. If you type 'f catch TAB RET', Info will complete the node name of the correctly written example and take you to the 'Catching Mistakes' node. (If you try this, you can return from the 'Catching Mistakes' node by typing 'l' ('Info-last').)

1.237 texi.guide/Debugging with TeX

Catching Errors with TeX Formatting

=====

You can also catch mistakes when you format a file with TeX.

Usually, you do this after you have run 'texinfo-format-buffer' (or, better, 'makeinfo-buffer') on the same file, because 'texinfo-format-buffer' sometimes displays error messages that make more sense than TeX. (See Debugging with Info, for more information.)

For example, TeX was run on a Texinfo file, part of which is shown here:

```
----- Buffer: texinfo.texi -----
name of the texinfo file as an extension. The
@samp{??} are 'wildcards' that cause the shell to
substitute all the raw index files. (@xref{sorting
indices, for more information about sorting
indices.})@refill
----- Buffer: texinfo.texi -----
```

(The cross reference lacks a closing brace.) TeX produced the following output, after which it stopped:

```
----- Buffer: *texinfo-tex-shell* -----
Runaway argument?
{sorting indices, for more information about sorting
```

```

indices.) @refill @ETC.
! Paragraph ended before @xref was complete.
<to be read again>
                                @par
1.27

?
----- Buffer: *texinfo-tex-shell* -----

```

In this case, TeX produced an accurate and understandable error message:

```
Paragraph ended before @xref was complete.
```

'@par' is an internal TeX command of no relevance to Texinfo. '1.27' means that TeX detected the problem on line 27 of the Texinfo file. The '?' is the prompt TeX uses in this circumstance.

Unfortunately, TeX is not always so helpful, and sometimes you must truly be a Sherlock Holmes to discover what went wrong.

In any case, if you run into a problem like this, you can do one of three things.

1. You can tell TeX to continue running and ignore just this error by typing RET at the '?' prompt.
2. You can tell TeX to continue running and to ignore all errors as best it can by typing 'r RET' at the '?' prompt.

This is often the best thing to do. However, beware: the one error may produce a cascade of additional error messages as its consequences are felt through the rest of the file. (To stop TeX when it is producing such an avalanche of error messages, type 'C-d' (or 'C-c C-d', if you are running a shell inside Emacs Version 18.))

3. You can tell TeX to stop this run by typing 'x RET' at the '?' prompt.

Please note that if you are running TeX inside Emacs, you need to switch to the shell buffer and line at which TeX offers the '?' prompt.

Sometimes TeX will format a file without producing error messages even though there is a problem. This usually occurs if a command is not ended but TeX is able to continue processing anyhow. For example, if you fail to end an itemized list with the '@end itemize' command, TeX will write a DVI file that you can print out. The only error message that TeX will give you is the somewhat mysterious comment that

```
(@end occurred inside a group at level 1)
```

However, if you print the DVI file, you will find that the text of the file that follows the itemized list is entirely indented as if it were part of the last item in the itemized list. The error message is the way TeX says that it expected to find an '@end' command somewhere in the file; but that it could not determine where it was needed.

Another source of notoriously hard-to-find errors is a missing '@end group' command. If you ever are stumped by incomprehensible errors, look for a missing '@end group' command first.

If the Texinfo file lacks header lines, TeX may stop in the beginning of its run and display output that looks like the following. The '*' indicates that TeX is waiting for input.

```
This is TeX, Version 2.0 for Berkeley UNIX
(preloaded format=plain-cm 87.10.25)
(test.texinfo [1])
*
```

In this case, simply type '\end RET' after the asterisk. Then write the header lines in the Texinfo file and run the TeX command again. (Note the use of the backslash, '\'. TeX uses '\' instead of '@'; and in this circumstance, you are working directly with TeX, not with Texinfo.)

1.238 texi.guide/Using texinfo-show-structure

Using 'texinfo-show-structure'
=====

It is not always easy to keep track of the nodes, chapters, sections, and subsections of a Texinfo file. This is especially true if you are revising or adding to a Texinfo file that someone else has written.

In GNU Emacs, in Texinfo mode, the 'texinfo-show-structure' command lists all the lines that begin with the @-commands that specify the structure: '@chapter', '@section', '@appendix', and so on. With an argument ('C-u' as prefix argument, if interactive), the command also shows the '@node' lines. The 'texinfo-show-structure' command is bound to 'C-c C-s' in Texinfo mode, by default.

The lines are displayed in a buffer called the '*Occur*' buffer. For example, when 'texinfo-show-structure' was run on an earlier version of this appendix, it produced the following:

```
Lines matching "^@\(chapter \|sect\|sub\|unnum\|major\|
heading \|appendix\)" in buffer texinfo.texi.
 4:@appendix Formatting Mistakes
52:@appendixsec Catching Errors with Info Formatting
222:@appendixsec Catching Errors with @TeX{} Formatting
338:@appendixsec Using @code{texinfo-show-structure}
407:@appendixsubsec Using @code{occur}
444:@appendixsec Finding Badly Referenced Nodes
513:@appendixsubsec Running @code{Info-validate}
573:@appendixsubsec Splitting a File Manually
```

This says that lines 4, 52, and 222 of 'texinfo.texi' begin with the '@appendix', '@appendixsec', and '@appendixsec' commands respectively.

If you move your cursor into the `*Occur*` window, you can position the cursor over one of the lines and use the `C-c C-c` command (`'occur-mode-goto-occurrence'`), to jump to the corresponding spot in the Texinfo file. See *Using Occur*, for more information about `'occur-mode-goto-occurrence'`.

The first line in the `*Occur*` window describes the "regular expression" specified by `TEXINFO-HEADING-PATTERN`. This regular expression is the pattern that `'texinfo-show-structure'` looks for. See *Using Regular Expressions*, for more information.

When you invoke the `'texinfo-show-structure'` command, Emacs will display the structure of the whole buffer. If you want to see the structure of just a part of the buffer, of one chapter, for example, use the `C-x n` (`'narrow-to-region'`) command to mark the region. (See *Narrowing*.) This is how the example used above was generated. (To see the whole buffer again, use `C-x w` (`'widen'`)).

If you call `'texinfo-show-structure'` with a prefix argument by typing `C-u C-c C-s`, it will list lines beginning with `'@node'` as well as the lines beginning with the `@`-sign commands for `'@chapter'`, `'@section'`, and the like.

You can remind yourself of the structure of a Texinfo file by looking at the list in the `*Occur*` window; and if you have mis-named a node or left out a section, you can correct the mistake.

1.239 texi.guide/Using occur

Using `'occur'`
=====

Sometimes the `'texinfo-show-structure'` command produces too much information. Perhaps you want to remind yourself of the overall structure of a Texinfo file, and are overwhelmed by the detailed list produced by `'texinfo-show-structure'`. In this case, you can use the `'occur'` command directly. To do this, type

```
'M-x occur'
```

and then, when prompted, type a "regexp", a regular expression for the pattern you want to match. (See *Regular Expressions*.) The `'occur'` command works from the current location of the cursor in the buffer to the end of the buffer. If you want to run `'occur'` on the whole buffer, place the cursor at the beginning of the buffer.

For example, to see all the lines that contain the word `'@chapter'` in them, just type `'@chapter'`. This will produce a list of the chapters. It will also list all the sentences with `'@chapter'` in the middle of the line.

If you want to see only those lines that start with the word `'@chapter'`, type `'^@chapter'` when prompted by `'occur'`. If you want to see all the lines that end with a word or phrase, end the last word

with a '\$'; for example, 'catching mistakes\$'. This can be helpful when you want to see all the nodes that are part of the same chapter or section and therefore have the same 'Up' pointer.

See Using Occur, for more information.

1.240 texi.guide/Running Info-Validate

Finding Badly Referenced Nodes

=====

You can use the 'Info-validate' command to check whether any of the 'Next', 'Previous', 'Up' or other node pointers fail to point to a node. This command checks that every node pointer points to an existing node. The 'Info-validate' command works only on Info files, not on Texinfo files.

The 'makeinfo' program validates pointers automatically, so you do not need to use the 'Info-validate' command if you are using 'makeinfo'. You only may need to use 'Info-validate' if you are unable to run 'makeinfo' and instead must create an Info file using 'texinfo-format-region' or 'texinfo-format-buffer', or if you write an Info file from scratch.

Using Info-validate	How to run 'Info-validate'.
Unsplit	How to create an unsplit file.
Tagifying	How to tagify a file.
Splitting	How to split a file manually.

1.241 texi.guide/Using Info-validate

Running 'Info-validate'

To use 'Info-validate', visit the Info file you wish to check and type:

```
M-x Info-validate
```

(Note that the 'Info-validate' command requires an upper case 'I'. You may also need to create a tag table before running 'Info-validate'. See Tagifying.)

If your file is valid, you will receive a message that says "File appears valid". However, if you have a pointer that does not point to a node, error messages will be displayed in a buffer called '*problems in info file*'.

For example, 'Info-validate' was run on a test file that contained

only the first node of this manual. One of the messages said:

In node "Overview", invalid Next: Texinfo Mode

This meant that the node called 'Overview' had a 'Next' pointer that did not point to anything (which was true in this case, since the test file had only one node in it).

Now suppose we add a node named 'Texinfo Mode' to our test case but we do not specify a 'Previous' for this node. Then we will get the following error message:

In node "Texinfo Mode", should have Previous: Overview

This is because every 'Next' pointer should be matched by a 'Previous' (in the node where the 'Next' points) which points back.

'Info-validate' also checks that all menu entries and cross references point to actual nodes.

Note that 'Info-validate' requires a tag table and does not work with files that have been split. (The 'texinfo-format-buffer' command automatically splits large files.) In order to use 'Info-validate' on a large file, you must run 'texinfo-format-buffer' with an argument so that it does not split the Info file; and you must create a tag table for the unsplit file.

1.242 texi.guide/Unsplit

Creating an Unsplit File

You can run 'Info-validate' only on a single Info file that has a tag table. The command will not work on the indirect subfiles that are generated when a master file is split. If you have a large file (longer than 70,000 bytes or so), you need to run the 'texinfo-format-buffer' or 'makeinfo-buffer' command in such a way that it does not create indirect subfiles. You will also need to create a tag table for the Info file. After you have done this, you can run 'Info-validate' and look for badly referenced nodes.

The first step is to create an unsplit Info file.

To prevent 'texinfo-format-buffer' from splitting a Texinfo file into smaller Info files, give a prefix to the 'M-x texinfo-format-buffer' command:

C-u M-x texinfo-format-buffer

or else

C-u C-c C-e C-b

When you do this, Texinfo will not split the file and will not create a

tag table for it.

1.243 texi.guide/Tagifying

Tagifying a File

After creating an unsplit Info file, you must create a tag table for it. Visit the Info file you wish to tagify and type:

M-x Info-tagify

(Note the upper case I in 'Info-tagify'.) This creates an Info file with a tag table that you can validate.

The third step is to validate the Info file:

M-x Info-validate

(Note the upper case I in 'Info-validate'.) In brief, the steps are:

C-u M-x texinfo-format-buffer

M-x Info-tagify

M-x Info-validate

After you have validated the node structure, you will be able to rerun 'texinfo-format-buffer' in the normal way so it will construct a tag table and split the file automatically, or you can make the tag table and split the file manually.

1.244 texi.guide/Splitting

Splitting a File Manually

You should split a large file or else let the 'texinfo-format-buffer' or 'makeinfo-buffer' command do it for you automatically. (Generally you will let one of the formatting commands do this job for you. See Create an Info File.)

The split-off files are called the indirect subfiles.

Info files are split to save memory. With smaller files, Emacs does not have make such a large buffer to hold the information.

If an Info file has more than 30 nodes, you should also make a tag table for it. See Using Info-validate, for information about creating a tag table. (Again, tag tables are usually created automatically by the formatting command; you only need to create a tag table yourself if you are doing the job manually. Most likely, you will do this for a large,

unsplit file on which you have run 'Info-validate'.)

Visit the Info file you wish to tagify and split and type the two commands:

```
M-x Info-tagify
M-x Info-split
```

(Note that the 'I' in 'Info' is upper case.)

When you use the 'Info-split' command, the buffer is modified into a (small) Info file which lists the indirect subfiles. This file should be saved in place of the original visited file. The indirect subfiles are written in the same directory the original file is in, with names generated by appending '-' and a number to the original file name.

The primary file still functions as an Info file, but it contains just the tag table and a directory of subfiles.

1.245 texi.guide/Refilling Paragraphs

Refilling Paragraphs

The '@refill' command refills and, optionally, indents the first line of a paragraph.⁽¹⁾ The '@refill' command is no longer important, but we describe it here because you once needed it. You will see it in many old Texinfo files.

Without refilling, paragraphs containing long @-constructs may look bad after formatting because the formatter removes @-commands and shortens some lines more than others. In the past, neither 'texinfo-format-region' nor 'texinfo-format-buffer' refilled paragraphs automatically. The '@refill' command had to be written at the end of every paragraph to cause these formatters to fill them. (Both TeX and 'makeinfo' have always refilled paragraphs automatically.) Now, all the Info formatters automatically fill and indent those paragraphs that need to be filled and indented.

The '@refill' command causes both the 'texinfo-format-region' command and the 'texinfo-format-buffer' command to refill a paragraph in the Info file *after* all the other processing has been done. For this reason, you can not use '@refill' with a paragraph containing either '@*' or '@w{ ... }' since the refilling action will override those two commands.

The 'texinfo-format-region' and 'texinfo-format-buffer' commands now automatically append '@refill' to the end of each paragraph that should be filled. They do not append '@refill' to the ends of paragraphs that contain '@*' or '@w{ ... }' and therefore do not refill or indent them.

----- Footnotes -----

(1) Perhaps the command should have been called the

'@refillandindent' command, but '@refill' is shorter and the name was chosen before indenting was possible.

1.246 texi.guide/Command Syntax

The character '@' is used to start special Texinfo commands. (It has the same meaning that '\' has in PlainTeX.) Texinfo has four types of @-command:

1. Non-alphabetic commands.

These commands consist of an @ followed by a punctuation mark or other character that is not part of the alphabet. Non-alphabetic commands are almost always part of the text within a paragraph, and never take any argument. The two characters (@ and the other one) are complete in themselves; none is followed by braces. The non-alphabetic commands are: '@.', '@:', '@*', '@@', '@}', '@}.'

2. Alphabetic commands that do not require arguments.

These commands start with @ followed by a word followed by left- and right-hand braces. These commands insert special symbols in the document; they do not require arguments. For example, '@dots{}' => '...', '@equiv{}' => '==', '@TeX{}' => 'TeX', and '@bullet{}' => '*'.

3. Alphabetic commands that require arguments within braces.

These commands start with @ followed by a letter or a word, followed by an argument within braces. For example, the command '@dfn' indicates the introductory or defining use of a term; it is used as follows: 'In Texinfo, @@-commands are @dfn{mark-up} commands.'

4. Alphabetic commands that occupy an entire line.

These commands occupy an entire line. The line starts with @, followed by the name of the command (a word); for example, '@center' or '@cindex'. If no argument is needed, the word is followed by the end of the line. If there is an argument, it is separated from the command name by a space. Braces are not used.

Thus, the alphabetic commands fall into classes that have different argument syntaxes. You cannot tell to which class a command belongs by the appearance of its name, but you can tell by the command's meaning: if the command stands for a glyph, it is in class 2 and does not require an argument; if it makes sense to use the command together with other text as part of a paragraph, the command is in class 3 and must be followed by an argument in braces; otherwise, it is in class 4 and uses the rest of the line as its argument.

The purpose of having a different syntax for commands of classes 3 and 4 is to make Texinfo files easier to read, and also to help the GNU Emacs paragraph and filling commands work properly. There is only one exception to this rule: the command '@refill', which is always used at

the end of a paragraph immediately following the final period or other punctuation character. '@refill' takes no argument and does **not** require braces. '@refill' never confuses the Emacs paragraph commands because it cannot appear at the beginning of a line.

1.247 texi.guide/Obtaining TeX

How to Obtain TeX

TeX is freely redistributable. You can obtain TeX for Unix systems from the University of Washington for a distribution fee.

To order a full distribution, send \$200.00 for a 1/2-inch 9-track 1600 bpi ('tar' or 'cpio') tape reel, or \$210.00 for a 1/4-inch 4-track QIC-24 ('tar' or 'cpio') cartridge, to:

Northwest Computing Support Center
DR-10, Thomson Hall 35
University of Washington
Seattle, Washington 98195

Please make checks payable to the University of Washington.

Prepaid orders are preferred but purchase orders are acceptable; however, purchase orders carry an extra charge of \$10.00, to pay for processing.

Overseas sites: please add to the base cost \$20.00 for shipment via air parcel post, or \$30.00 for shipment via courier.

Please check with the Northwest Computing Support Center at the University of Washington for current prices and formats:

telephone: (206) 543-6259
email: elisabet@u.washington.edu

1.248 texi.guide/New Features

Second Edition Features

The second edition of the Texinfo manual describes more than 20 new Texinfo mode commands and more than 50 previously undocumented Texinfo edition.

Here is a brief description of the new commands.

New Texinfo Mode Commands The updating commands are especially useful.

New Commands

Many newly described @-commands.

1.249 texi.guide/New Texinfo Mode Commands

New Texinfo Mode Commands

=====

Texinfo mode provides commands and features especially designed for working with Texinfo files. More than 20 new commands have been added, including commands for automatically creating and updating both nodes and menus. This is a tedious task when done by hand.

The keybindings are intended to be somewhat mnemonic.

Update all nodes and menus

The `'texinfo-master-menu'` command is the primary command:

`'C-c C-u m'`

`'M-x texinfo-master-menu'`

Create or update a master menu. With `'C-u'` as a prefix argument, first create or update all nodes and regular menus.

Update Pointers

Create or update `'Next'`, `'Previous'`, and `'Up'` node pointers.

See Updating Nodes and Menus.

`'C-c C-u C-n'`

`'M-x texinfo-update-node'`

Update a node.

`'C-c C-u C-e'`

`'M-x texinfo-every-node-update'`

Update every node in the buffer.

Update Menus

Create or update menus.

See Updating Nodes and Menus.

`'C-c C-u C-m'`

`'M-x texinfo-make-menu'`

Make or update a menu.

`'C-c C-u C-a'`

`'M-x texinfo-all-menus-update'`

Make or update all the menus in a buffer. With `'C-u'` as a prefix argument, first update all the nodes.

Insert Title as Description

Insert a node's chapter or section title in the space for the description in a menu entry line; position point so you can edit the insert. (This command works somewhat differently than the other insertion commands, which insert only a predefined string.)

See Inserting Frequently Used Commands.

```
'C-c C-c C-d'
  Insert title.
```

Format for Info

Provide keybindings both for the Info formatting commands that are written in Emacs Lisp and for 'makeinfo' that is written in C.

See Info Formatting.

Use the Emacs lisp 'texinfo-format...' commands:

```
'C-c C-e C-r'
  Format the region.
```

```
'C-c C-e C-b'
  Format the buffer.
```

Use 'makeinfo':

```
'C-c C-m C-r'
  Format the region.
```

```
'C-c C-m C-b'
  Format the buffer.
```

```
'C-c C-m C-l'
  Recenter the 'makeinfo' output buffer.
```

```
'C-c C-m C-k'
  Kill the 'makeinfo' formatting job.
```

Typeset and Print

Typeset and print Texinfo documents from within Emacs.

See Printing.

```
'C-c C-t C-r'
  Run TeX on the region.
```

```
'C-c C-t C-b'
  Run TeX on the buffer.
```

```
'C-c C-t C-i'
  Run 'texindex'.

'C-c C-t C-p'
  Print the DVI file.

'C-c C-t C-q'
  Show the print queue.

'C-c C-t C-d'
  Delete a job from the print queue.

'C-c C-t C-k'
  Kill the current TeX formatting job.

'C-c C-t C-x'
  Quit a currently stopped TeX formatting job.

'C-c C-t C-l'
  Recenter the output buffer.
```

Other Updating Commands

The "other updating commands" do not have standard keybindings because they are used less frequently.

See Other Updating Commands.

```
'M-x texinfo-insert-node-lines'
  Insert missing '@node' lines using section titles as node names.

'M-x texinfo-multiple-files-update'
  Update a multi-file document. With a numeric prefix, such as 'C-u
  8', update *every* pointer and menu in *all* the files and then
  insert a master menu.

'M-x texinfo-indent-menu-description'
  Indent descriptions in menus.

'M-x texinfo-sequential-node-update'
  Insert node pointers in strict sequence.
```

1.250 texi.guide/New Commands

New Texinfo @-Commands

=====

The second edition of the Texinfo manual describes more than 50 commands that were not described in the first edition. A third or so of these commands existed in Texinfo but were not documented in the manual; the others are new. Here is a listing, with brief descriptions of them:

Indexing

Create your own index, and merge indices.

See Indices.

`'@defindex INDEX-NAME'`

Define a new index and its indexing command. See also the `'@defcodeindex'` command.

`'@synindex FROM-INDEX INTO-INDEX'`

Merge the FROM-INDEX index into the INTO-INDEX index. See also the `'@syncodeindex'` command.

Definitions

Describe functions, variables, macros, commands, user options, special forms, and other such artifacts in a uniform format.

See Definition Commands.

`'@defn CATEGORY NAME ARGUMENTS...'`

Format a description for functions, interactive commands, and similar entities.

`'@defvr, @defop, ...'`

15 other related commands.

Glyphs

Indicate the results of evaluation, expansion, printed output, an error message, equivalence of expressions, and the location of point.

See Glyphs.

`'@equiv{}'`

`'=='`

Equivalence:

`'@error{}'`

`'error-->'`

Error message

`'@expansion{}'`

`'==>'`

Macro expansion

`'@point{}'`

`'-!-'`

Position of point

`'@print{}'`

`'-|'`

Printed output

```
`@result{}'
`=>'
    Result of an expression
```

Page Headings

Customize page headings.

See Headings.

```
`@headings ON-OFF-SINGLE-DOUBLE'
    Headings on or off, single, or double-sided.
```

```
`@evenfooting [LEFT] @| [CENTER] @| [RIGHT]'
    Footings for even-numbered (left-hand) pages.
```

```
`@evenheading, @everyheading, @oddheading, ...'
    Five other related commands.
```

```
`@thischapter'
    Insert name of chapter and chapter number.
```

```
`@thischaptername, @thisfile, @thistitle, @thispage'
    Related commands.
```

Formatting

Format blocks of text.

See Quotations and Examples, and
See Making Lists and Tables.

```
`@cartouche'
    Draw rounded box surrounding text (not in Info).
```

```
`@enumerate OPTIONAL-ARG'
    Enumerate a list with letters or numbers.
```

```
`@exdent LINE-OF-TEXT'
    Remove indentation.
```

```
`@flushleft'
    Left justify.
```

```
`@flushright'
    Right justify.
```

```
`@format'
    Do not narrow nor change font.
```

```
`@ftable FORMATTING-COMMAND'
```

```
`@vtable FORMATTING-COMMAND'
    Two-column table with indexing.
```

`'@lisp'`

For an example of Lisp code.

`'@smallexample'`

`'@smalllisp'`

Like `@table` and `@lisp` but for `@smallbook`.

Conditionals

Conditionally format text.

See `'@set'` `'@clear'` `'@value'`.

`'@set FLAG [STRING]'`

Set a flag. Optionally, set value of FLAG to STRING.

`'@clear FLAG'`

Clear a flag.

`'@value{FLAG}'`

Replace with value to which FLAG is set.

`'@ifset FLAG'`

Format, if FLAG is set.

`'@ifclear FLAG'`

Ignore, if FLAG is set.

Produce unnumbered headings that do not appear in a table of contents.

See Structuring.

`'@heading TITLE'`

Unnumbered section-like heading not listed in the table of contents of a printed manual.

`'@chapheading, @majorheading, @subheading, @subsubheading'`

Related commands.

Font commands

See Smallcaps, and

See Fonts.

`'@r{TEXT}'`

Print in roman font.

`'@sc{TEXT}'`

Print in SMALL CAPS font.

Miscellaneous

See See '@title' '@subtitle' and '@author' Commands,
 see See Overfull hboxes,
 see See Footnotes,
 see See Format a Dimension,
 see See Inserting a Minus Sign,
 see See Paragraph Indenting,
 see See Cross Reference Commands,
 see See '@title' '@subtitle' and '@author', and
 see See How to Make Your Own Headings.

'@author AUTHOR'

Typeset author's name.

'@finalout'

Produce cleaner printed output.

'@footnotestyle'

Specify footnote style.

'@dmn{DIMENSION}'

Format a dimension.

'@minus{'

Generate a minus sign.

'@paragraphindent'

Specify paragraph indentation.

'@ref{NODE-NAME, [ENTRY], [TOPIC-OR-TITLE], [INFO-FILE], [MANUAL]]'

Make a reference. In the printed manual, the reference does not start with the word 'see'.

'@title TITLE'

Typeset TITLE in the alternative title page format.

'@subtitle SUBTITLE'

Typeset SUBTITLE in the alternative title page format.

'@today{'

Insert the current date.

1.251 texi.guide/Command and Variable Index

Command and Variable Index

This is an alphabetical list of all the @-commands and several variables. To make the list easier to use, the commands are listed without their preceding '@'.

* (force line break)
 . (true end of sentence)

Line Breaks
 Controlling Spacing

:	(suppress widening)	Controlling Spacing
@	(single @)	Inserting An Atsign
{	(single {)	Inserting Braces
}	(single)	Inserting Braces
a	fourpaper	A4 Paper
a	appendix	unnumbered & appendix
a	appendixsec	unnumberedsec appendixsec heading
a	appendixsection	unnumberedsec appendixsec heading
a	appendixsubsec	unnumberedsubsec appendixsubsec subheading
a	appendixsubsubsec	subsubsection
a	apply	Sample Function Definition
a	author	title subtitle author
b	(bold font)	Fonts
b	buffer-end	Def Cmd Template
b	bullet	bullet
b	bye	Ending a File
b	bye	File End
c	(comment)	Comments
c	cartouche	cartouche
c	center	titlefont center sp
c	chapheading	majorheading & chapheading
c	chapter	chapter
c	cindex	Indexing Commands
c	cite	cite
c	clear	ifset ifclear
c	code	code
c	comment	Comments
c	contents	Contents
c	copyright	copyright symbol
c	copyright	Copyright & Permissions
c	cropmarks	Cropmarks and Magnification
c	defcodeindex	New Indices
c	defcv	Abstract Objects
c	deffn	Functions Commands
c	deffnx	deffnx
c	defindex	New Indices
c	defivar	Abstract Objects
c	defmac	Functions Commands
c	defmethod	Abstract Objects
c	defop	Abstract Objects
c	defopt	Variables Commands
c	defspec	Functions Commands
c	deftp	Data Types
c	deftypefn	Typed Functions
c	deftypefun	Typed Functions
c	deftypevar	Typed Variables
c	deftypevr	Typed Variables
c	defun	Functions Commands
c	defvar	Variables Commands
c	defvr	Variables Commands
c	dfn	dfn
c	display	display
c	dmn	dmn
c	dots	dots
c	emph	emph & strong
c	end	Quotations and Examples
c	end	Introducing Lists

end titlepage	end titlepage
enumerate	enumerate
evenfooting	Custom Headings
evenheading	Custom Headings
everyfooting	Custom Headings
everyheading	Custom Headings
example	example
exdent	exdent
file	file
filll	Copyright & Permissions
finalout	Overfull hboxes
findex	Indexing Commands
flushleft	flushleft & flushright
flushright	flushleft & flushright
foobar	Optional Arguments
footnote	Footnotes
footnotestyle	Footnotes
format	format
forward-word	Def Cmd Template
ftable	ftable vtable
group	group
heading	unnumberedsec appendixsec heading
headings	headings on off
i (italic font)	Fonts
ifclear	ifset ifclear
ifinfo	Conditionals
ifset	ifset ifclear
iftex	Conditionals
ignore	Comments
include	Using Include Files
Info-validate	Running Info-Validate
INFOPATH	Other Info Directories
inforef	inforef
input (TeX command)	Minimum
isearch-backward	deffnx
isearch-forward	deffnx
item	itemize
item	table
itemize	itemize
itemx	itemx
kbd	kbd
key	key
kindex	Indexing Commands
lisp	Lisp Example
lpr (DVI print command)	Shell Format & Print
mag (TeX command)	Cropmarks and Magnification
majorheading	majorheading & chapheading
makeinfo-buffer	makeinfo in Emacs
makeinfo-kill-job	makeinfo in Emacs
makeinfo-recenter-output-buffer	makeinfo in Emacs
makeinfo-region	makeinfo in Emacs
menu	Menus
minus	minus
need	need
next-error	makeinfo in Emacs
noindent	noindent
occur	Using occur

occur-mode-goto-occurrence	Showing the Structure
oddfooting	Custom Headings
oddheading	Custom Headings
page	page
page-delimiter	Showing the Structure
paragraphindent	paragraphindent
pindex	Indexing Commands
printindex	Printing Indices & Menus
pxref	pxref
quotation	quotation
r (Roman font)	Fonts
ref	ref
refill	Refilling Paragraphs
samp	samp
sc (small caps font)	Smallcaps
section	section
set	ifset ifclear
setchapternewpage	setchapternewpage
setfilename	setfilename
settitle	settitle
shortcontents	Contents
smallbook	smallbook
smallexample	smallexample & smalllisp
smalllisp	smallexample & smalllisp
sp (line spacing)	sp
sp (titlepage line spacing)	titlefont center sp
strong	emph & strong
subheading	unnumberedsubsec appendixsubsec subheading
subsection	subsection
subsubheading	subsubsection
subsubsection	subsubsection
subtitle	title subtitle author
summarycontents	Contents
syncodeindex	syncodeindex
syncodeindex	syncodeindex
synindex	synindex
t (typewriter font)	Fonts
table	Two-column Tables
tex	Using Ordinary TeX Commands
tex (command)	tex
texi2dvi (shell script)	Shell Format & Print
texindex	Format-Print Hardcopy
texindex	Shell Format & Print
texinfo-all-menus-update	Updating Commands
texinfo-every-node-update	Updating Commands
texinfo-format-buffer	Info Formatting
texinfo-format-buffer	texinfo-format commands
texinfo-format-buffer	texinfo-format commands
texinfo-format-region	texinfo-format commands
texinfo-format-region	texinfo-format commands
texinfo-format-region	Info Formatting
texinfo-indent-menu-description	Other Updating Commands
texinfo-insert-@code	Inserting
texinfo-insert-@dfn	Inserting
texinfo-insert-@end	Inserting
texinfo-insert-@example	Inserting
texinfo-insert-@item	Inserting

texinfo-insert-@kbd	Inserting
texinfo-insert-@node	Inserting
texinfo-insert-@noindent	Inserting
texinfo-insert-@samp	Inserting
texinfo-insert-@table	Inserting
texinfo-insert-@var	Inserting
texinfo-insert-braces	Inserting
texinfo-insert-node-lines	Other Updating Commands
texinfo-make-menu	Updating Commands
texinfo-master-menu	Updating Commands
texinfo-multiple-files-update	texinfo-multiple-files-update
texinfo-multiple-files-update (in brief)	Other Updating Commands
texinfo-sequential-node-update	Other Updating Commands
texinfo-show-structure	Using texinfo-show-structure
texinfo-show-structure	Showing the Structure
texinfo-start-menu-description	Inserting
texinfo-tex-buffer	Printing
texinfo-tex-print	Printing
texinfo-tex-region	Printing
texinfo-update-node	Updating Commands
TEXINPUTS	Preparing for TeX
thischapter	Custom Headings
thischaptername	Custom Headings
thisfile	Custom Headings
thispage	Custom Headings
thistitle	Custom Headings
tindex	Indexing Commands
title	title subtitle author
titlefont	titlefont center sp
titlepage	titlepage
today	Custom Headings
top (@-command)	makeinfo top command
unnumbered	unnumbered & appendix
unnumberedsec	unnumberedsec appendixsec heading
unnumberedsubsec	unnumberedsubsec appendixsubsec subheading
unnumberedsubsubsec	subsubsection
up-list	Inserting
value	value
var	var
vindex	Indexing Commands
vskip	Copyright & Permissions
vtable	ftable vtable
w (prevent line break)	w
xref	xref

1.252 texi.guide/Concept Index

Concept Index

@-command in nodename
@-command list

Node Line Requirements
Command List

@-command syntax	Command Syntax
@-commands	Formatting Commands
.cshrc initialization file	Preparing for TeX
.profile initialization file	Preparing for TeX
@include file sample	Sample Include File
@menu parts	Menu Parts
@node line writing	Writing a Node
makeinfo inside Emacs	makeinfo in Emacs
makeinfo options	makeinfo options
TEXINPUTS environment variable	Preparing for TeX
dir directory for Info installation	Install an Info File
dir file listing	New Info File
End node footnote style	Footnotes
Separate footnote style	Footnotes
Top node	The Top Node
Top node is first	First Node
Top node naming for references	Top Node Naming
Top node summary	Top Node Summary
hboxes, overfull	Overfull hboxes
ifinfo permissions	ifinfo Permissions
TeX commands, using ordinary	Using Ordinary TeX Commands
TeX index sorting	Format-Print Hardcopy
TeX input initialization	Preparing for TeX
TeX, how to obtain	Obtaining TeX
A4 paper, printing on	A4 Paper
Abbreviations for keys	key
Adding a new info file	New Info File
Alphabetical @-command list	Command List
Another Info directory	Other Info Directories
Apostrophe in nodename	Node Line Requirements
Arguments, repeated and optional	Optional Arguments
Automatic pointer creation with makeinfo	makeinfo Pointer Creation
Automatically insert nodes, menus	Updating Nodes and Menus
Badly referenced nodes	Running Info-Validate
Batch formatting for Info	Batch Formatting
Beginning a Texinfo file	Beginning a File
Beginning line of a Texinfo file	First Line
Black rectangle in hardcopy	Overfull hboxes
Blank lines	sp
Book characteristics, printed	Printed Books
Book, printing small	smallbook
Box with rounded corners	cartouche
Braces and argument syntax	Command Syntax
Braces, inserting	Braces Atsigns Periods
Braces, when to use	Formatting Commands
Breaks in a line	Line Breaks
Buffer formatting and printing	Printing
Bullets, inserting	Dots Bullets
Capitalizing index entries	Indexing Commands
Case in nodename	Node Line Requirements
Catching errors with TeX formatting	Debugging with TeX
Catching errors with Info formatting	Debugging with Info
Catching mistakes	Catching Mistakes
Chapter structuring	Structuring
Characteristics, printed books or manuals	Printed Books
Checking for badly referenced nodes	Running Info-Validate
Colon in nodename	Node Line Requirements

Combining indices	Combining Indices
Comma in nodename	Node Line Requirements
Command definitions	Sample Function Definition
Commands to insert single characters	Braces Atsigns Periods
Commands using ordinary TeX	Using Ordinary TeX Commands
Commands, inserting them	Inserting
Comments	Comments
Compile command for formatting	Compile-Command
Conditionally visible text	Conditionals
Conditions for copying Texinfo	Copying
Contents, Table of	Contents
Contents-like outline of file structure	Showing the Structure
Conventions for writing definitions	Def Cmd Conventions
Conventions, syntactic	Conventions
Copying conditions	Copying
Copying permissions	Sample Permissions
Copying software	Software Copying Permissions
Copyright page	Copyright & Permissions
Correcting mistakes	Catching Mistakes
Create nodes, menus automatically	Updating Nodes and Menus
Creating an Info file	Create an Info File
Creating an unsplit file	Unsplit
Creating index entries	Indexing Commands
Creating indices	Indices
Creating pointers with makeinfo	makeinfo Pointer Creation
Cropmarks for printing	Cropmarks and Magnification
Cross reference parts	Cross Reference Parts
Cross references	Cross References
Cross references using @inforef	inforef
Cross references using @pxref	pxref
Cross references using @ref	ref
Cross references using @xref	xref
Debugging the Texinfo structure	Catching Mistakes
Debugging with TeX formatting	Debugging with TeX
Debugging with Info formatting	Debugging with Info
Defining indexing entries	Indexing Commands
Defining new indices	New Indices
Definition commands	Definition Commands
Definition conventions	Def Cmd Conventions
Definition template	Def Cmd Template
Definitions grouped together	deffnx
Description for menu, start	Inserting
Different cross reference commands	Cross Reference Commands
Dimension formatting	dmn
Display formatting	display
Distribution	Software Copying Permissions
Dots, inserting	Dots Bullets
Dots, inserting	dots
Double-colon menu entries	Less Cluttered Menu Entry
DVI file	Shell Format & Print
Ellipsis, inserting	Dots Bullets
Emacs	Texinfo Mode
Emacs shell, format, print from	Within Emacs
Emphasizing text	Emphasis
Emphasizing text, font for	emph & strong
End of header line	End of Header
End titlepage starts headings	end titlepage

Ending a Texinfo file	Ending a File
Entries for an index	Indexing Commands
Entries, making index	Index Entries
Enumeration	enumerate
Equivalence, indicating it	Equivalence
Error message, indicating it	Error Glyph
Errors, parsing	makeinfo in Emacs
European A4 paper	A4 Paper
Evaluation glyph	result
Example for a small book	smallexample & smalllisp
Example menu	Menu Example
Examples, formatting them	example
Expansion, indicating it	expansion
File beginning	Beginning a File
File ending	Ending a File
File section structure, showing it	Showing the Structure
Filling paragraphs	Refilling Paragraphs
Final output	Overfull hboxes
Finding badly referenced nodes	Running Info-Validate
First line of a Texinfo file	First Line
First node	First Node
Fonts for indices	syncodeindex
Fonts for printing, not for Info	Fonts
Footings	Headings
Footnotes	Footnotes
Format a dimension	dmn
Format and print hardcopy	Format-Print Hardcopy
Format and print in Texinfo mode	Texinfo Mode Printing
Format with the compile command	Compile-Command
Format, print from Emacs shell	Within Emacs
Formatting a file for Info	Create an Info File
Formatting commands	Formatting Commands
Formatting examples	example
Formatting for Info	Info Formatting
Formatting for printing	Printing
Formatting headings and footings	Headings
Formatting requirements	Requirements Summary
Frequently used commands, inserting	Inserting
Function definitions	Sample Function Definition
General syntactic conventions	Conventions
Generating menus with indices	Printing Indices & Menus
Glyphs	Glyphs
GNU Emacs	Texinfo Mode
GNU Emacs shell, format, print from	Within Emacs
Going to other Info files' nodes	Other Info Files
Group (hold text together vertically)	group
Grouping two definitions together	deffnx
Hardcopy, printing it	Format-Print Hardcopy
Header for Texinfo files	Header
Header of a Texinfo file	First Line
Headings	Headings
Headings, page, begin to appear	end titlepage
Highlighting text	Indicating
Hints	Tips
Holding text together vertically	group
If text conditionally visible	Conditionals
Ignored text	Comments

Include file requirements	Include File Requirements
Include file sample	Sample Include File
Include files	Include Files
Indentation undoing	exdent
Indenting paragraphs	paragraphindent
Index entries	Indexing Commands
Index entries, making	Index Entries
Index entry capitalization	Indexing Commands
Index font types	Indexing Commands
Indexing commands, predefined	Indexing Commands
Indexing table entries automatically	ftable vtable
Indicating commands, definitions, etc.	Indicating
Indicating evaluation	result
Indices	Indices
Indices, combining them	Combining Indices
Indices, defining new	New Indices
Indices, printing and menus	Printing Indices & Menus
Indices, sorting	Format-Print Hardcopy
Indices, two letter names	syncodeindex
Indirect subfiles	Tag and Split Files
Info batch formatting	Batch Formatting
Info file installation	Install an Info File
Info file requires @setfilename	setfilename
Info file, listing new one	New Info File
Info file, splitting manually	Splitting
Info files	Info Files
Info formatting	Info Formatting
Info installed in another directory	Other Info Directories
Info validating a large file	Using Info-validate
Info, creating an on-line file	Create an Info File
Info; other files' nodes	Other Info Files
Initialization file for TeX input	Preparing for TeX
Insert nodes, menus automatically	Updating Nodes and Menus
Inserting @, braces, and periods	Braces Atsigns Periods
Inserting dots	Dots Bullets
Inserting dots	dots
Inserting ellipsis	Dots Bullets
Inserting frequently used commands	Inserting
Inserting special characters and symbols	Insertions
Installing an Info file	Install an Info File
Installing Info in another directory	Other Info Directories
Introduction, as part of file	Software Copying Permissions
Itemization	itemize
Keys, recommended names	key
Larger or smaller pages	Cropmarks and Magnification
Less cluttered menu entry	Less Cluttered Menu Entry
License agreement	Software Copying Permissions
Line breaks	Line Breaks
Line breaks, preventing	w
Line spacing	sp
Lisp example	Lisp Example
Lisp example for a small book	smallexample & smalllisp
List of @-commands	Command List
Listing a new info file	New Info File
Lists and tables, making them	Lists and Tables
Local variables	Compile-Command
Location of menus	Menu Location

Looking for badly referenced nodes	Running Info-Validate
Macro definitions	Sample Function Definition
Magnified printing	Cropmarks and Magnification
Making a printed manual	Format-Print Hardcopy
Making a tag table automatically	Tag and Split Files
Making a tag table manually	Unsplit
Making cross references	Cross References
Making line and page breaks	Breaks
Making lists and tables	Lists and Tables
Manual characteristics, printed	Printed Books
Marking text within a paragraph	Marking Text
Marking words and phrases	Marking Text
Master menu	The Top Node
Master menu parts	Master Menu Parts
Mathematical expressions	Using Ordinary TeX Commands
Menu description, start	Inserting
Menu entries with two colons	Less Cluttered Menu Entry
Menu example	Menu Example
Menu location	Menu Location
Menu parts	Menu Parts
Menu writing	Writing a Menu
Menus	Menus
Menus generated with indices	Printing Indices & Menus
META key	key
Meta-syntactic chars for arguments	Optional Arguments
Minimal Texinfo file (requirements)	Minimum
Mistakes, catching	Catching Mistakes
Mode, using Texinfo	Texinfo Mode
Must have in Texinfo file	Minimum
Names for indices	syncodeindex
Names recommended for keys	key
Naming a 'Top' Node in references	Top Node Naming
Need space at page bottom	need
New index defining	New Indices
New info file, listing it in dir file	New Info File
Node line requirements	Node Line Requirements
Node line writing	Writing a Node
Node, defined	node
Node, 'Top'	The Top Node
Nodename must be unique	Node Line Requirements
Nodename, cannot contain	Node Line Requirements
Nodes for menus are short	Menu Location
Nodes in other Info files	Other Info Files
Nodes, catching mistakes	Catching Mistakes
Nodes, checking for badly referenced	Running Info-Validate
Obtaining TeX	Obtaining TeX
Occurrences, listing with @occur	Using occur
Optional and repeated arguments	Optional Arguments
Options for makeinfo	makeinfo options
Ordinary TeX commands, using	Using Ordinary TeX Commands
Other Info files' nodes	Other Info Files
Outline of file structure, showing it	Showing the Structure
Overfull hboxes	Overfull hboxes
Overview of Texinfo	Overview
Page breaks	page
Page delimiter in Texinfo mode	Showing the Structure
Page headings	Headings

Page numbering	Headings
Page sizes for books	smallbook
Pages, starting odd	setchapternewpage
Paper size, European A4	A4 Paper
Paragraph indentation	paragraphindent
Paragraph, marking text within	Marking Text
Parsing errors	makeinfo in Emacs
Part of file formatting and printing	Printing
Parts of a cross reference	Cross Reference Parts
Parts of a master menu	Master Menu Parts
Parts of a menu	Menu Parts
Periods, inserting	Braces Atsigns Periods
Permissions	Sample Permissions
Permissions, printed	Copyright & Permissions
PlainTeX	Using Ordinary TeX Commands
Point, indicating it in a buffer	Point Glyph
Pointer creation with makeinfo	makeinfo Pointer Creation
Pointer validation with makeinfo	Pointer Validation
Predefined indexing commands	Indexing Commands
Predefined names for indices	syncodeindex
Preparing to use TeX	Preparing for TeX
Preventing line and page breaks	Breaks
Print and format in Texinfo mode	Texinfo Mode Printing
Print, format from Emacs shell	Within Emacs
Printed book and manual characteristics	Printed Books
Printed output, indicating it	Print Glyph
Printed permissions	Copyright & Permissions
Printing a region or buffer	Printing
Printing an index	Printing Indices & Menus
Printing cropmarks	Cropmarks and Magnification
Problems, catching	Catching Mistakes
Quotations	quotation
Recommended names for keys	key
Rectangle, ugly, black in hardcopy	Overfull hboxes
References	Cross References
References using @inforef	inforef
References using @pxref	pxref
References using @ref	ref
References using @xref	xref
Referring to other Info files	Other Info Files
Refilling paragraphs	Refilling Paragraphs
Region formatting and printing	Printing
Region printing in Texinfo mode	Texinfo Mode Printing
Repeated and optional arguments	Optional Arguments
Required in Texinfo file	Minimum
Requirements for formatting	Requirements Summary
Requirements for include files	Include File Requirements
Requirements for updating commands	Updating Requirements
Result of an expression	result
Running Info-validate	Using Info-validate
Running makeinfo in Emacs	makeinfo in Emacs
Running an Info formatter	Info Formatting
Sample @include file	Sample Include File
Sample function definition	Sample Function Definition
Sample Texinfo file	Short Sample
Sample Texinfo file, no comments	Sample Texinfo File
Section structure of a file, showing it	Showing the Structure

Shell, format, print from	Within Emacs
Shell, running makeinfo in	makeinfo in Emacs
Short nodes for menus	Menu Location
Showing the section structure of a file	Showing the Structure
Showing the structure of a file	Using texinfo-show-structure
Single characters, commands to insert	Braces Atsigns Periods
Size of printed book	smallbook
Small book example	smallexample & smalllisp
Small book size	smallbook
Small caps font	Smallcaps
Software copying permissions	Software Copying Permissions
Sorting indices	Format-Print Hardcopy
Spaces (blank lines)	sp
Special insertions	Insertions
Special typesetting commands	Dots Bullets
Specifying index entries	Indexing Commands
Splitting an Info file manually	Splitting
Start of header line	Start of Header
Starting chapters	setchapternewpage
Structure of a file, showing it	Showing the Structure
Structure, catching mistakes in	Catching Mistakes
Structuring of chapters	Structuring
Subsection-like commands	unnumberedsubsec appendixsubsec subheading
Subsub commands	subsubsection
Syntactic conventions	Conventions
Syntax, optional & repeated arguments	Optional Arguments
Table of contents	Contents
Tables and lists, making them	Lists and Tables
Tables with indexes	ftable vtable
Tables, making two-column	Two-column Tables
Tabs; don't use!	Conventions
Tag table, making automatically	Tag and Split Files
Tag table, making manually	Unsplit
Template for a definition	Def Cmd Template
Texinfo file beginning	Beginning a File
Texinfo file ending	Ending a File
Texinfo file header	Header
Texinfo file minimum	Minimum
Texinfo file section structure, showing it	Showing the Structure
Texinfo mode	Texinfo Mode
Texinfo overview	Overview
Texinfo printed book characteristics	Printed Books
Text, conditionally visible	Conditionals
Thin space between number, dimension	dmn
Tips	Tips
Title page	titlepage
Titlepage end starts headings	end titlepage
Titlepage permissions	Titlepage Permissions
Tree structuring	Tree Structuring
Two letter names for indices	syncodeindex
Two named items for @table	itemx
Two part menu entry	Less Cluttered Menu Entry
Two 'First' Lines for @deffn	deffnx
Typesetting commands for dots, etc.	Dots Bullets
Uncluttered menu entry	Less Cluttered Menu Entry
Unique nodename requirement	Node Line Requirements
Unprocessed text	Comments

Unsplit file creation	Unsplit
Updating nodes and menus	Updating Nodes and Menus
Updating requirements	Updating Requirements
Usage tips	Tips
Validating a large file	Using Info-validate
Validation of pointers	Pointer Validation
Value of an expression, indicating	result
Vertical whitespace (vskip)	Copyright & Permissions
Vertically holding text together	group
Visibility of conditional text	Conditionals
Words and phrases, marking them	Marking Text
Writing a menu	Writing a Menu
Writing an @node line	Writing a Node
