makeinfo

COLLABORATORS			
	TITLE : makeinfo		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 7, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

makeinfo

1.1 makeinfo.guide

```
What is makeinfo?
*****************

makeinfo is a program for converting TeXinfo files into Info files or
AmigaGuide® hypertext files.  TeXinfo is a documentation system
that uses a single source file to produce both on-line information and
printed output.

You can read the on-line information using Info; type info to learn
about Info.  See Texinfo, to learn about the TeXinfo documentation
system.
```

1.2 makeinfo.guide/Formatting Control

```
Controlling Paragraph Formats
=============================

Without any special options, makeinfo fills the paragraphs that it
outputs to an Info file.  Filling is the process of breaking 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.  With makeinfo, you can control:

   * The width of each paragraph (the fill-column).

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

1.3 makeinfo.guide/Options

```
Command Line Options
====================

The following command line options are available for makeinfo.

-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.  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
     Supress 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.

--amiga
     Converts a TeXinfo file to AmigaGuide® V34 hypertext format.

--amiga-39
     Converts a TeXinfo file to AmigaGuide® V39 hypertext format.

--amiga-40
     Converts a TeXinfo file to AmigaGuide® V40 hypertext format.

--index-button-length NUMBER
     An index button will have a minimum length of NUMBER. The default
     value for index-button-length is 40. (Only used if converting to
     an AmigaGuide® file).

--menu-button-length NUMBER
     A menu button will have a minimum length of NUMBER. The default
     value for menu-button-length is 25. (Only used if converting to an
     AmigaGuide® file).

--help
     Show a summary of the commend line arguments to makeinfo.
```

1.4 makeinfo.guide/Pointer Validation

```
Pointer Validation
==================

If you do not suppress pointer-validation (by using the
--no-pointer-validation option), makeinfo will check the validity
of the final Info file.  Mostly, this means ensuring that nodes you
have referenced really exist.  Here is a complete list of what is
checked:

  1. If a ‘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.
```

1.5 makeinfo.guide/The Macro Facility

```
The Macro Facility
==================

This chapter describes the new macro facility.

A macro  is a command that you define in terms of other commands.  It
doesn't exist as a TeXinfo command until you define it as part of the
input file to Makeinfo.  Once the command exists, it behaves much as
any other TeXinfo command.  Macros are a useful way to ease the details
and tedium of writing a 'correct' info file.  The following sections
explain how to write and invoke macros.
```

1.6 makeinfo.guide/How to Use Macros in TeXinfo

```
How to Use Macros in TeXinfo
----------------------------

Using macros in TeXinfo is easy.  First you define the macro.  After
that, the macro command is available as a normal TeXinfo command.  Here
is what a definition looks like:

     @macro NAME{ARG1, ... ARGN}
     TEXINFO COMMANDS...
     @end macro

The arguments that you specify that the macro takes are expanded with
the actual parameters used when calling the macro if they are seen
surrounded by backslashes.  For example, here is a definition of
which surrounds its argument with  @code{...}.

     @macro codeitem{item}
     @item @code{\item\}
     @end macro
```

When the macro is expanded, all of the text between the @macro and
the actual parameters substituted for the named parameters. So, a call
to the above macro might look like:

```
     @codeitem{Foo}
```

and Makeinfo would execute the following code:

```
     @item @code{Foo}
```

A special case is made for macros which only take a single argument, and
which are invoked without any brace characters (i.e., { ...})
surrounding an argument; the rest of the line is supplied as is as the
sole argument to the macro. This special case allows one to redefine
some standard TeXinfo commands without modifying the input file. Along
with the non-recursive action of macro invocation, one can easily
redefine the sectioning commands to also provide index entries:

```
     @macro chapter{name}
     @chapter \name\
     @findex \name\
     @end macro
```

Thus, the text:

```
     @chapter strlen
```

will expand to:

```
     @chapter strlen
     @findex strlen
```

1.7 makeinfo.guide/Using Macros Recursively

Using Macros Recursively

Normally, while a particular macro is executing, any call to that macro
will be seen as a call to a builtin TeXinfo command. This allows one
to redefine a builtin TeXinfo command as a macro, and then use that
command within the definition of the macro itself. For example, one
might wish to make sure that whereever a term was defined with
index for the manual. Here is a macro which redefines @dfn to do just
that:

```
     @macro dfn{text}
     @dfn{\text\}
     @cpindex \text\
     @end macro
```

Note that we used the builtin TeXinfo command @dfn within our
overriding macro definition.

This behaviour itself can be overridden for macro execution by writing a

special macro control command in the definition of the macro. The
command is considered special because it doesn't affect the output text
directly, rather, it affects the way in which the macro is defined. One
such special command is @allow-recursion.

```
     @macro silly{arg}
     @allow-recursion
     \arg\
     @end macro
```

Now @silly is a macro that can be used within a call to itself:

```
     This text @silly{@silly{some text}} is ``some text''.
```

1.8 makeinfo.guide/Using TeXinfo Macros As Arguments

Using TeXinfo Macros As Arguments

How to use TeXinfo macros as arguments to other TeXinfo macros.

1.9 makeinfo.guide/Index

Index
=====