Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'ctags.1p' command

$ man ctags.1p

CTAGS(1P) POSIX Programmer's Manual CTAGS(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

ctags ? create a tags file (DEVELOPMENT, FORTRAN)

SYNOPSIS

ctags [-a] [-f tagsfile] pathname...

ctags -x pathname...

DESCRIPTION

The ctags utility shall be provided on systems that support the the Software Development Utilities option, and either or both of the C-Lan? guage Development Utilities option and FORTRAN Development Utilities option. On other systems, it is optional.

The ctags utility shall write a tagsfile or an index of objects from C-language or FORTRAN source files specified by the pathname operands.

The tagsfile shall list the locators of language-specific objects within the source files. A locator consists of a name, pathname, and either a search pattern or a line number that can be used in searching for the object definition. The objects that shall be recognized are specified in the EXTENDED DESCRIPTION section.

OPTIONS

The ctags utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-a Append to tagsfile.

-f tagsfile

Write the object locator lists into tagsfile instead of the default file named tags in the current directory.

-x Produce a list of object names, the line number, and filename in which each is defined, as well as the text of that line, and write this to the standard output. A tagsfile shall not be created when -x is specified.

OPERANDS

The following pathname operands are supported:

file.c Files with basenames ending with the .c suffix shall be treated as C-language source code. Such files that are not valid input to c99 produce unspecified results.

file.h Files with basenames ending with the .h suffix shall be treated as C-language source code. Such files that are not valid input to c99 produce unspecified results.

file.f Files with basenames ending with the .f suffix shall be treated as FORTRAN-language source code. Such files that are not valid input to fort77 produce unspecified results.

The handling of other files is implementation-defined.

STDIN

See the INPUT FILES section.

INPUT FILES

The input files shall be text files containing source code in the lan? guage indicated by the operand filename suffixes.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of ctags:

LANG Provide a default value for the internationalization vari?

ables that are unset or null. (See the Base Definitions vol?
ume of POSIX.1?2017, Section 8.2, Internationalization Vari?
ables for the precedence of internationalization variables
used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE
Determine the order in which output is sorted for the -x op?
tion. The POSIX locale determines the order in which the tagsfile is written.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files). When processing C-language source code, if the locale is not compatible with the C locale described by the ISO C standard, the results are unspecified.

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er?
ror.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The list of object name information produced by the -x option shall be written to standard output in the following format:

"%s %d %s %s", <object-name>, <line-number>, <filename>, <text>

where <text> is the text of line <line-number> of file <filename>.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

When the -x option is not specified, the format of the output file

shall be:

"%s\t%s\t/%s/\n", <identifier>, <filename>, <pattern>

where <pattern> is a search pattern that could be used by an editor to find the defining instance of <identifier> in <filename> (where defin? ing instance is indicated by the declarations listed in the EXTENDED DESCRIPTION).

An optional <circumflex> ('^') can be added as a prefix to <pattern>, and an optional <dollar-sign> can be appended to <pattern> to indicate that the pattern is anchored to the beginning (end) of a line of text.

Any <slash> or <backslash> characters in <pattern> shall be preceded by a <backslash> character. The anchoring <circumflex>, <dollar-sign>, and escaping <backslash> characters shall not be considered part of the search pattern. All other characters in the search pattern shall be considered literal characters.

An alternative format is:

"%s\t%s\t?%s?\n", <identifier>, <filename>, <pattern>

which is identical to the first format except that <slash> characters in <pattern> shall not be preceded by escaping <backslash> characters, and <question-mark> characters in <pattern> shall be preceded by <back? slash> characters.

A second alternative format is:

"%s\t%s\t%d\n", <identifier>, <filename>, <lineno>

where <lineno> is a decimal line number that could be used by an editor to find <identifier> in <filename>.

Neither alternative format shall be produced by ctags when it is used as described by POSIX.1?2008, but the standard utilities that process tags files shall be able to process those formats as well as the first format.

In any of these formats, the file shall be sorted by identifier, based on the collation sequence in the POSIX locale.

EXTENDED DESCRIPTION

If the operand identifies C-language source, the ctags utility shall attempt to produce an output line for each of the following objects:

* Function definitions
* Type definitions
* Macros with arguments

It may also produce output for any of the following objects:

* Function prototypes
* Structures
* Unions
* Global variable definitions
* Enumeration types
* Macros without arguments
* #define statements
* #line statements

Any #if and #ifdef statements shall produce no output. The tag main is treated specially in C programs. The tag formed shall be created by prefixing M to the name of the file, with the trailing .c, and leading pathname components (if any) removed.

On systems that do not support the C-Language Development Utilities option, ctags produces unspecified results for C-language source code files. It should write to standard error a message identifying this condition and cause a non-zero exit status to be produced.

If the operand identifies FORTRAN source, the ctags utility shall produce an output line for each function definition. It may also produce output for any of the following objects:

* Subroutine definitions
* COMMON statements
* PARAMETER statements
* DATA and BLOCK DATA statements
* Statement numbers

On systems that do not support the FORTRAN Development Utilities option, ctags produces unspecified results for FORTRAN source code files. It should write to standard error a message identifying this condition and cause a non-zero exit status to be produced.

It is implementation-defined what other objects (including duplicate

identifiers) produce output.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

The output with -x is meant to be a simple index that can be written out as an off-line readable function index. If the input files to ctags (such as .c files) were not created using the same locale as that in effect when ctags -x is run, results might not be as expected.

The description of C-language processing says ``attempts to'' because the C language can be greatly confused, especially through the use of #defines, and this utility would be of no use if the real C preproces? sor were run to identify them. The output from ctags may be fooled and incorrect for various constructs.

EXAMPLES

None.

RATIONALE

The option list was significantly reduced from that provided by histor? ical implementations. The -F option was omitted as redundant, since it is the default. The -B option was omitted as being of very limited use? fulness. The -t option was omitted since the recognition of typedefs is now required for C source files. The -u option was omitted because the update function was judged to be not only inefficient, but also rarely needed.

An early proposal included a -w option to suppress warning diagnostics. Since the types of such diagnostics could not be described, the option was omitted as being not useful.

The text for LC_CTYPE about compatibility with the C locale acknowl? edges that the ISO C standard imposes requirements on the locale used

to process C source. This could easily be a superset of that known as ``the C locale'' by way of implementation extensions, or one of a few alternative locales for systems supporting different codesets. No statement is made for FORTRAN because the ANSI X3.9?1978 standard (FOR? TRAN 77) does not (yet) define a similar locale concept. However, a general rule in this volume of POSIX.1?2017 is that any time that lo? cales do not match (preparing a file for one locale and processing it in another), the results are suspect.

The collation sequence of the tags file is not affected by LC_COLLATE because it is typically not used by human readers, but only by programs such as vi to locate the tag within the source files. Using the POSIX locale eliminates some of the problems of coordinating locales between the ctags file creator and the vi file reader.

Historically, the tags file has been used only by ex and vi. However, the format of the tags file has been published to encourage other pro? grams to use the tags in new ways. The format allows either patterns or line numbers to find the identifiers because the historical vi recog? nizes either. The ctags utility does not produce the format using line numbers because it is not useful following any source file changes that add or delete lines. The documented search patterns match historical practice. It should be noted that literal leading <circumflex> or trailing <dollar-sign> characters in the search pattern will only be? have correctly if anchored to the beginning of the line or end of the line by an additional <circumflex> or <dollar-sign> character.

Historical implementations also understand the objects used by the lan? guages Pascal and sometimes LISP, and they understand the C source out? put by lex and yacc. The ctags utility is not required to accommodate these languages, although implementors are encouraged to do so.

The following historical option was not specified, as vgrind is not in? cluded in this volume of POSIX.1?2017:

-v If the -v flag is given, an index of the form expected by vgrind is produced on the standard output. This listing con? tains the function name, filename, and page number (assuming

64-line pages). Since the output is sorted into lexicographic order, it may be desired to run the output through sort -f.

Sample use:

```
ctags -v files | sort -f > index vgrind -x index
```

The special treatment of the tag main makes the use of ctags practical in directories with more than one program.

FUTURE DIRECTIONS

None.

SEE ALSO

c99, fort77, vi

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

COPYRIGHT