IEEE P1003.2a Draft 8 – December 1991

History of Decisions Made

The `−n` option was added to extend the range of filenames that could be handled.

Consideration was given to adding a `−a` flag to use the alphabetic file name generation used by the historical `split` utility, but the functionality added by the `−n` option was deemed to make alphabetic naming unnecessary.

5.7 `ctags` — Create a tags file

This utility shall be provided on systems that support the User Portability Utili- 7
ties Option, the Software Development Utilities Option, and either or both of the 7
C Language Development Utilities Option and FORTRAN Development Utilities 7
Option. On other systems it is optional. 7

5.7.1 Synopsis

`ctags [−ax][−f` *tagsfile* `]` *pathname* ...

5.7.2 Description

The `ctags` utility shall create a *tags* file from C-language or FORTRAN source file(s) specified by the *pathname* operands. The tags file shall list the locators of language-specific objects within the source files. A locator consists of a name, pathname, and either basic regular expression or a line number that can be used in searching for the object definition. The objects that shall be recognized are specified in 5.7.7.

5.7.3 Options

The `ctags` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`−a` Append to tags file.

`−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 file name in which each is defined, as well as the text of that line, and write this to the standard output.

5.7.4 Operands

The following *pathname* operands shall be supported by the implementation:

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 7
`c89` produce unspecified results. 7

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 7
`c89` produce unspecified results. 7

file `.f` Files with basenames ending with the `.f` suffix shall be treated
as FORTRAN language source code. Such files that are not valid 7
input to `fort77` produce unspecified results. 7

The handling of other files is implementation defined.

5.7.5 External Influences

5.7.5.1 Standard Input

See Input Files.

5.7.5.2 Input Files

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

5.7.5.3 Environment Variables

The following environment variables shall affect the execution of `ctags`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_COLLATE This variable shall determine the order in which output is
sorted for the `−x` option. The POSIX Locale shall deter- 8
mine the order in which the tags file is written. 8

LC_CTYPE This variable shall determine the interpretation of
sequences of bytes of text data as characters (e.g., single-
versus multibyte characters within arguments and input
files). When processing C-language source code, if the 7

locale is not compatible with the C locale described by the C Standard {7}, the results are unspecified.

LC_MESSAGES This variable shall determine the language in which messages should be written.

5.7.5.4 Asynchronous Events

Default.

5.7.6 External Effects

5.7.6.1 Standard Output

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>*.

5.7.6.2 Standard Error

Used only for diagnostic messages.

5.7.6.3 Output Files

The format of the output file shall be:

`"%s\t%s\t/%s/\n"`, *<identifier>*, *<filename>*, *<rexp>*

where *<rexp>* is a basic regular expression (see 2.8.3) that could be used by an editor to find the defining instance of *<identifier>* in *<filename>* (where ‘‘defining instance’’ is indicated by the declarations listed in 5.7.7).

An 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>*. This alternative format shall not be produced by `ctags` when it is used as described by this standard, but the standard utilities that process tags files shall be able to process this format as well as the preceding one.

In either format, the file shall be sorted by identifier, based on the collation sequence in the POSIX Locale.

5.7.7 Extended Description

If the operand identifies C-language source, the `ctags` utility shall attempt to 7
produce an output line for each of the following objects: 7

— Function definitions 7

— Typedefs 7

— Macros with arguments 7

It may also produce output for any of the following objects: 7

— Function prototypes 7

— Structures 7

— Unions 7

— Global variable definitions 7

— Enums 7

— Macros without arguments 7

— `#define` statements 7

— `#line` statements 7

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 7
Utilities Option in Annex A, `ctags` produces unspecified results for C-language 7
source code files. It should write to standard error a message identifying this con- 7
dition and cause a nonzero exit status to be produced. 7

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 in 7
Annex C, `ctags` produces unspecified results for FORTRAN source code files. It 7
should write to standard error a message identifying this condition and cause a 7
nonzero exit status to be produced. 7

It is implementation defined what other objects (including duplicate identifiers) produce output.

5.7.8 Exit Status

The `ctags` utility shall exit with one of the following values:

0 Successful completion.

>0 An error occurred.

5.7.9 Consequences of Errors

Default.

5.7.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

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

History of Decisions Made

The option list was significantly reduced from that provided by historical implementations. The `−F` option was omitted as redundant, since it is the default. The `−B` option was omitted as being of very limited usefulness. 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 earlier draft 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 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 locales as those in effect when `ctags −x` is run, 8
results might not be as expected. 8

The description of C-language processing says ‘‘shall attempt to’’ because the C 7
language can be greatly obfuscated, especially through the use of `#defines`, and 7
this utility would be of no use if the real C preprocessor were run to identify them. 7
The output from `ctags` may be fooled and incorrect for various constructs. 7

The text for **LC_CTYPE** about compatibility with the C locale acknowledges that 7
the C Standard {7} imposes requirements on the locale used to process C source. 7
This could easily be a superset of that known as ‘‘the C locale’’ by way of imple- 7
mentation extensions, or one of a few alternative locales for systems supporting 7
different code sets. No statement is made for FORTRAN because ISO 1539 {2} 7
(FORTRAN 77) does not (yet) define a similar locale concept. However, a general 7
rule in POSIX.2 is that any time locales do not match (preparing a file for one 7
locale and processing it in another), results are suspect. 7

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 programs to use the tags in new ways. The format allows either regular expressions or line numbers to find the identifiers because the historical `vi` recognizes either. The `ctags` utility does not produce the format using line numbers because it are not useful following any source file changes that add or delete lines.

Historical implementations also understand the objects used by the languages Pascal and sometimes LISP, and understand the C source output 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 included in this standard.

`−v` If the `−v` flag is given, an index of the form expected by `vgrind` is produced on the standard output. This listing contains the function name, file name, and page number (assuming 64 line pages). Since the output will be 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
```

5.8 `df` — Report free disk space

5.8.1 Synopsis

`df [−kP]` [*file* . . .]

5.8.2 Description

The `df` utility shall write the amount of available space for file systems on which the invoking user has appropriate read access. File systems shall be specified by the *file* operands; when none are specified, information shall be written for all file systems. The format of the default output from `df` is unspecified, but all space figures shall be reported in 512-byte units, unless the `−k` option is specified. This output shall contain at least the file system names and amount of available space on each of these file systems.

5.8.3 Options

The `df` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`−k` Use 1024-byte units, instead of the default 512-byte units, when writing space figures.

`−P` Produce output in the format described in 5.8.6.1.

5.8.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file within the hierarchy of the desired file system. If a file other than a FIFO , a regular file, or a directory is specified, the results are unspecified. Otherwise, `df` shall write the amount of free space in the file system containing the specified *file* operand.

5.8.5 External Influences

5.8.5.1 Standard Input

None.