-- --

IEEE P1003.2 Draft 11.2 – September 1991

4.14 `cut` — Cut out selected fields of each line of a file

4.14.1 Synopsis

`cut −b` *list* [`−n`] [*file* . . .]

`cut −c` *list* [*file* . . .]

`cut −f` *list* [`−d` *delim*] [`−s`] [*file* . . .]

4.14.2 Description

The `cut` utility shall cut out bytes (`−b` option), characters (`−c` option), or character-delimited fields (`−f` option) from each line in one or more files, concatenate them, and write them to standard output.

4.14.3 Options

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

The option-argument *list* (see options `−b`, `−c`, and `−f` below) shall be a comma-separated list or `<blank>`-separated list of positive numbers and ranges. Ranges can be in three forms. The first is two positive numbers separated by a hyphen (*low*−*high*), which represents all fields from the first number to the second number. The second is a positive number preceded by a hyphen (−*high*), which represents all fields from field number 1 to that number. The third is a positive number followed by a hyphen (*low*−), which represents that number to the last field, inclusive. The elements in list can be repeated, can overlap, and can be specified in any order.

The following options shall be supported by the implementation:

`−b` *list* Cut based on a *list* of bytes. Each selected byte shall be output unless the `−n` option is also specified. It shall not be an error to select bytes not present in the input line.

`−c` *list* Cut based on a *list* of characters. Each selected character shall be output. It shall not be an error to select characters not present in the input line.

`−d` *delim* Set the field delimiter to the character *delim* . The default is the `<tab>` character.

`−f` *list* Cut based on a *list* of fields, assumed to be separated in the file by a delimiter character (see `−d`). Each selected field shall be output. Output fields shall be separated by a single occurrence of the field delimiter character. Lines with no field delimiters shall be passed through intact, unless `−s` is specified. It shall not be an error to select fields not present in the input line.

`−n` Do not split characters. When specified with the `−b` option, each element in *list* of the form *low−high* (hyphen-separated numbers) shall be modified as follows:

> If the byte selected by *low* is not the first byte of a character, *low* shall be decremented to select the first byte of the character originally selected by *low*. If the byte selected by *high* is not the last byte of a character, *high* shall be decremented to select the last byte of the character prior to the character originally selected by *high*, or zero if there is no prior character. If the resulting range element has *high* equal to zero or *low* greater than *high*, the list element shall be dropped from *list* for that input line without causing an error.

Each element in list of the form *low−* shall be treated as above with *high* set to the the number of bytes in the current line, not including the terminating `<newline>` character. Each element in list of the form *−high* shall be treated as above with *low* set to 1. Each element in list of the form *num* (a single number) shall be treated as above with *low* set to *num* and *high* set to *num*.

`−s` Suppress lines with no delimiter characters, when used with the `−f` option. Unless specified, lines with no delimiters shall be passed through untouched.

4.14.4 Operands

The following operands shall be supported by the implementation:

file A pathname of an input file. If no *file* operands are specified, or if a *file* operand is −, the standard input shall be used.

4.14.5 External Influences

4.14.5.1 Standard Input

The standard input shall be used only if no *file* operands are specified, or if a *file* operand is −. See Input Files.

4.14.5.2 Input Files

The input files shall be text files, except that line lengths shall be unlimited.

4.14.5.3 Environment Variables

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

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_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files).

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

4.14.5.4 Asynchronous Events

Default.

4.14.6 External Effects

4.14.6.1 Standard Output

The `cut` utility output shall be a concatenation of the selected bytes, characters, or fields (one of the following):

`"%s\n"`, *<concatenation of bytes>*

`"%s\n"`, *<concatenation of characters>*

`"%s\n"`, *<concatenation of fields and field delimiters>*

4.14.6.2 Standard Error

Used only for diagnostic messages.

4.14.6.3 Output Files

None.

4.14.7 Extended Description

None.

4.14.8 Exit Status

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

0	All input files were output successfully.
>0	An error occurred.

4.14.9 Consequences of Errors

Default.

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

Examples, Usage

Examples of the option qualifier list:

`1,4,7`	Select the first, fourth, and seventh bytes, characters, or fields and field delimiters.
`1-3,8`	Equivalent to `1,2,3,8`.
`-5,10`	Equivalent to `1,2,3,4,5,10`.
`3-`	Equivalent to third through last.

The *low*−*high* forms are not always equivalent when used with `−b` and `−n` and 1
multibyte characters. See the description of `−n`. 1

The following command:

```
cut -d : -f 1,6 /etc/passwd
```

reads the System V password file (user database) and produces lines of the form:

<user ID>:*<home directory>*

Most utilities in this standard work on text files. The `cut` utility can be used to turn files with arbitrary line lengths into a set of text files containing the same data. The `paste` utility can be used to create (or recreate) files with arbitrary line lengths. For example, if `file` contains long lines:

```
cut -b 1-500 -n file > file1
cut -b 501- -n file > file2
```

creates `file1` (a text file) with lines no longer than 500 bytes (plus the `<newline>` character and `file2` that contains the remainder of the data from file. (Note that `file2` will not be a text file if there are lines in `file` that are longer than 500 + {LINE_MAX} bytes.) The original file can be recreated from `file1` and file2 using the command:

```
paste -d "\0" file1 file2 > file
```

History of Decisions Made

Some historical implementations do not count <backspace> characters in determining character counts with the −c option. This may be useful for using cut for processing nroff output. It was deliberately decided not to have the −c option treat either <backspace> or <tab> characters in any special fashion. The fold 1
utility does treat these characters specially. 1

Unlike other utilities, some historical implementations of cut exit after not finding an input file, rather than continuing to process the remaining *file* operands. This behavior is prohibited by this standard, where only the exit status is affected by this problem.

The behavior of cut when provided with either mutually exclusive options or options that do not make sense together has been deliberately left unspecified in favor of global wording in Section 2.

The traditional cut utility has worked in an environment where bytes and characters were equivalent (modulo <backspace> and <tab> processing in some implementations). In the extended world of multibyte characters, the new −b option has been added. The −n option (used with −b) allows it to be used to act on bytes rounded to character boundaries. The algorithm specified for −n guarantees that

```
cut -b 1-500 -n file > file1
cut -b 501- -n file > file2
```

will end up with all the characters in file appearing exactly once in file1 or file2. (There is, however, a <newline> character in both file1 and file2 for each <newline> character in file.)

4.15 `date` — Write the date and time

4.15.1 Synopsis

`date [−u] [+`*format*`]`

4.15.2 Description

The `date` utility shall write the date and time to standard output. By default, the current date and time shall be written. If an operand beginning with `+` is specified, the output format of `date` shall be controlled by the field descriptors and other text in the operand.

4.15.3 Options

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

The following option shall be supported by the implementation:

`−u`	Perform operations as if the **TZ** environment variable was set to the string `UTC0`, or its equivalent historical value of `GMT0`. Otherwise, `date` shall use the time zone indicated by the **TZ** environment variable or the system default if that variable is not set. 2

4.15.4 Operands

When the format is specified, each field descriptor shall be replaced in the standard output by its corresponding value. All other characters shall be copied to the output without change. The output shall be always terminated with a `<newline>` character.

Field Descriptors

`%a`	Locale's abbreviated weekday name.
`%A`	Locale's full weekday name.
`%b`	Locale's abbreviated month name.
`%B`	Locale's full month name.
`%c`	Locale's appropriate date and time representation.
`%C`	Century (a year divided by 100 and truncated to an integer) as a decimal number (00-99).
`%d`	Day of the month as a decimal number (01-31).
`%D`	Date in the format *mm/dd/yy* .
`%e`	Day of the month as a decimal number (1-31 in a two-digit field with leading `<space>` fill).
`%h`	A synonym for `%b`.