IEEE P1003.2 Draft 11.2 – September 1991

one 24-hour period in the past, not any time from the beginning of that period to 1
the current time. For example, `–atime 3` is true if the file was accessed any time 1
in the period from 72 to 48 hours ago. 1

Historical implementations do not modify `{}` when it appears as a substring of an `–exec` or `–ok` *utility_name* or argument string. There have been numerous user requests for this extension, so this standard allows the desired behavior. At least one recent implementation does support this feature, but ran into several problems in managing memory allocation and dealing with multiple occurrences of `{}` in a string while it was being developed, so it is not yet required behavior.

Assuming the presence of `–print` was added at the request of several working group members to correct a historical pitfall that plagues novice users. It is entirely upward compatible from the historical System V `find` utility and should be easy to implement. In its simplest form (`find` *directory*), it could be confused with the historical BSD fast `find`. The BSD developers agree that adding `–print` as a default expression is the right thing to do and believe that the fast `find` functionality should have been/should be provided by a separate utility. They suggest that the new utility be called `locate`.

4.25 `fold` — Fold lines

4.25.1 Synopsis

`fold [–bs] [–w` *width* `] [`*file* `...]`

4.25.2 Description

The `fold` utility is a filter that shall fold lines from its input files, breaking the lines to have a maximum of *width* column positions (or bytes, if the `–b` option is specified). Lines shall be broken by the insertion of a `<newline>` character such that each output line (referred to later in this clause as a segment) is the maximum width possible that does not exceed the specified number of column positions (or bytes). A line shall not be broken in the middle of a character. The behavior is undefined if *width* is less than the number of columns any single character in the input would occupy.

If the `<carriage-return>`, `<backspace>`, or `<tab>` characters are encountered 2
in the input, and the `–b` option is not specified, they shall be treated specially:

`<carriage-return>` 2
: The current count of line width shall be set to zero. The `fold` utility 2
shall not insert a `<newline>` immediately before or after any 2
`<carriage-return>`. 2

`<backspace>`
: The current count of line width shall be decremented by one, although the count never shall become negative. The `fold` utility shall not

insert a `<newline>` immediately before or after any `<backspace>`.

`<tab>` Each `<tab>` character encountered shall advance the column position pointer to the next tab stop. Tab stops shall be at each column position *n* such that *n* modulo 8 equals 1.

4.25.3 Options

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

The following options shall be supported by the implementation:

`−b` Count *width* in bytes rather than column positions.

`−s` If a segment of a line contains a `<blank>` within the first *width* column positions (or bytes), break the line after the last such `<blank>` meeting the width constraints. If there is no `<blank>` meeting the requirements, the `−s` option shall have no effect for that output segment of the input line.

`−w` *width* Specify the maximum line length, in column positions (or bytes if `−b` is specified). The results are unspecified if *width* is not a positive decimal number. The default value shall be 80.

4.25.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a text file to be folded. If no *file* operands are specified, the standard input shall be used.

4.25.5 External Influences

4.25.5.1 Standard Input

The standard input shall be used only if no *file* operands are specified. See Input Files.

4.25.5.2 Input Files

If the `−b` option is specified, the input files shall be text files except that the lines are not limited to {LINE_MAX} bytes in length. If the `−b` option is not specified, the input files shall be text files.

4.25.5.3 Environment Variables

The following environment variables shall affect the execution of fold:

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) and for the determination of the width in column positions each character would occupy on a constant-width-font output device.

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

4.25.5.4 Asynchronous Events

Default.

4.25.6 External Effects

4.25.6.1 Standard Output

The standard output shall be a file containing a sequence of characters whose order shall be preserved from the input file(s), possibly with inserted <newline> characters.

4.25.6.2 Standard Error

Used only for diagnostic messages.

4.25.6.3 Output Files

None.

4.25.7 Extended Description

None.

4.25.8 Exit Status

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

- 0 All input files were processed successfully.
- >0 An error occurred.

4.25.9 Consequences of Errors

Default.

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

Examples, Usage

The `cut` and `fold` utilities can be used to create text files out of files with arbitrary line lengths. The `cut` utility should be used when the number of lines (or records) needs to remain constant. The `fold` utility should be used when the contents of long lines needs to be kept contiguous.

The `fold` utility is frequently used to send text files to line printers that truncate, rather than fold, lines wider than the printer is able to print (usually 80 or 132 column positions.)

Although terminal input in canonical processing mode requires the erase character (frequently set to `<backspace>`) to erase the previous character (not byte or column position), terminal output is not buffered and is extremely difficult, if not impossible, to parse correctly; the interpretation depends entirely on the physical device that will actually display/print/store the output. In all known internationalized implementations, the utilities producing output for mixed column width output assume that a `<backspace>` backs up one column position and outputs enough `<backspace>`s to get back to the start of the character when `<backspace>` is used to provide local line motions to support underlining and emboldening operations. Since `fold` without the `–b` option is dealing with these same constraints, `<backspace>` is always treated as backing up one column position rather than backing up one character.

An example invocation that submits a file of possibly long lines to the line printer (under the assumption that the user knows the line width of the printer to be assigned by `lp`):

```
fold -w 132 bigfile | lp
```

History of Decisions Made

Historical versions of the `fold` utility assumed one byte was one character and occupied one column position when written out. This is no longer always true. Since the most common usage of `fold` is believed to be folding long lines for output to limited-length output devices, this capability was preserved as the default case. The `-b` option was added so that applications could `fold` files with arbitrary length lines into text files that could then be processed by the utilities in this standard. Note that although the width for the `-b` option is in bytes, a line will never be split in the middle of a character. (It is unspecified what happens if a width is specified that is too small to hold a single character found in the input followed by a `<newline>`.)

The use of a hyphen as an option to specify standard input was removed from an earlier draft because it adds no functionality and is not historical practice.

The tab stops are hardcoded to be every eighth column to meet historical practice. No new method of specifying other tab stops was invented.

4.26 getconf — Get configuration values

4.26.1 Synopsis

`getconf` *system_var*

`getconf` *path_var* *pathname*

4.26.2 Description

In the first synopsis form, the `getconf` utility shall write to the standard output the value of the variable specified by the *system_var* operand.

In the second synopsis form, the `getconf` utility shall write to the standard output the value of the variable specified by the *path_var* operand for the path specified by the *pathname* operand.

The value of each configuration variable shall be determined as if it were obtained by calling the function from which it is defined to be available by this standard or by POSIX.1 {8} (see Operands). The value shall reflect conditions in the current operating environment.