IEEE P1003.2a Draft 8 – December 1991

5.27 `tabs` — Set terminal tabs

5.27.1 Synopsis

`tabs` [−*n*] [−`T` *type*]

`tabs` [−`T` *type*] *n1* [`,` *n2* `,`. . .]

5.27.2 Description

The `tabs` utility shall display a series of characters that first clears the hardware terminal tab settings and then initializes the tab stops at the specified positions.

The phrase ''tab-stop position *N*'' shall be taken to mean that, from the start of a line of output, tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position on that line. The maximum number of tab stops allowed is terminal dependent.

It need not be possible to implement `tabs` on certain terminals. If the terminal type obtained from the **TERM** environment variable or −`T` option represents such a terminal, an appropriate diagnostic message shall be written to standard error and `tabs` shall exit with a status greater than zero.

5.27.3 Options

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

The following options shall be supported by the implementation:

−*n* Specify repetitive tab stops separated by a uniform number of column positions, *n*, where *n* is a single-digit decimal number. The default usage of `tabs` with no arguments shall be equivalent to `tabs −8`. When `−0` is used, the tab stops shall be cleared and no new ones set.

−`T` *type* Indicate the type of terminal. If this option is not supplied and
the **TERM** variable is unset or null, an unspecified default termi- 8
nal type shall be used. The setting of *type* shall take precedence 8
over the value in **TERM**.

5.27.4 Operands

The following operand shall be supported by the implementation:

n1 [`,` *n2* `,`. . .]

A single command-line argument that consists of tab-stop values separated using either commas or `<blank>`s. The tab-stop values shall be positive decimal integers in strictly ascending order. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. For example, the tab lists `1,10,20,30` and `1,10,+10,+10` are considered to be identical.

5.27.5 External Influences

5.27.5.1 Standard Input

None.

5.27.5.2 Input Files

None.

5.27.5.3 Environment Variables

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

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).

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

TERM This variable shall be interpreted as the name of the terminal type. If this variable is unset or null, and if the `−T` 7
option is not specified, an unspecified default terminal 7
type shall be used. 7

5.27.5.4 Asynchronous Events

Default.

5.27.6 External Effects

5.27.6.1 Standard Output

If standard output is a terminal, the appropriate sequence to clear and set the tab stops may be written to standard output in an unspecified format. If standard output is not a terminal, undefined results occur.

5.27.6.2 Standard Error

Used only for diagnostic messages.

5.27.6.3 Output Files

None.

5.27.7 Extended Description

None.

5.27.8 Exit Status

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

0	Successful completion.
>0	An error occurred.

5.27.9 Consequences of Errors

Default.

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

Usage, Examples

Some integrated display units might not have escape sequences to set tab stops, but may be set by internal system calls. On these terminals, `tabs` will work if standard output is directed to the terminal; if output is directed to another file, however, `tabs` will fail.

History of Decisions Made

Consideration was given to having the `tput` utility handle all of the functions described here. However, the separate `tabs` utility was retained because it seems more intuitive to use a command named `tabs` than `tput` with a new option. The POSIX. 2a `tput` does not support setting or clearing tabs and no known historical version of `tabs` supports the capability of setting arbitrary tab stops.

The System V `tabs` interface is very complex; the version in this standard has a reduced feature list. There was considerable sentiment for specifying only a means of resetting the tabs back to a known state—presumably the ‘‘standard’’ tabs every eight positions. The following features were omitted:

(1) Setting tab stops tailored for certain programming languages; the working group was afraid that there would be no end to the bickering about which languages to include and where the tabs should be (even C programmers cannot agree on this).

(2) Setting tab stops via the first line in a file, using `--`*file*. Since even the *SVID* has no complete explanation of this feature, it is doubtful that it is in widespread use.

(3) Setting the left margin using `"+m`*n*`"`. As this does not work with all terminal types, it was omitted.

In an earlier draft, a `-t` *tablist* option was added for consistency with `expand`; this was later removed when inconsistencies with the historical list of tabs were identified.

Consideration was given to adding a `-p` option that would output the current tab settings so that they could be saved and then later restored. This was not accepted because querying the terminal’s tab stops is not a capability in historical *terminfo* or *termcap* facilities and might not be supportable on a wide range of terminals.

5.28 `talk` — Talk to another user

5.28.1 Synopsis

`talk` *address* [*terminal*]

5.28.2 Description

The `talk` utility is a two-way, screen-oriented communication program.

When first invoked, `talk` shall send a message similar to:

`Message from` <*unspecified string*>
`talk: connection requested by` *your_address*
`talk: respond with: talk` *your_address*

to the specified *address* . At this point, the recipient of the message can reply by typing:

`talk` *your_address*

Once communication is established, the two parties can type simultaneously, with
their output displayed in separate regions of the screen. Characters shall be pro- 8
cessed as follows: 8

- Typing the `<alert>` character shall alert the recipient's terminal.
- Typing `control-L` shall cause the sender's screen regions to be refreshed.
- Typing the erase and kill characters shall affect the sender's terminal in the manner described by the POSIX.1 {8} *termios* interface.
- Typing the interrupt or end-of-file characters shall terminate the local `talk` utility. Once the `talk` session has been terminated on one side, the other side of the `talk` session shall be notified that the `talk` session has been terminated and shall be able to do nothing except exit.
- Typing characters from LC_CTYPE classifications `print` or `space` shall 8
  cause those characters to be sent to the recipient's terminal.
- When and only when the `stty` *iexten* local mode (see 4.59.4.4) is enabled, 8
  additional special control characters and multibyte or single-byte functions 8
  shall be processed in an implementation-defined manner. (See POSIX.1 {8} 8
  section 7.1.1.9.) 8
- Typing other nonprintable characters shall cause implementation-defined sequences of printable characters to be sent to the recipient's terminal.

Permission to be a recipient of a `talk` message can be denied or granted by use of
the `mesg` utility (see 5.17). However, a user's privilege may further constrain the 7
domain of accessibility of other users' terminals. The `talk` utility shall fail when 7
the user lacks the appropriate privileges to perform the requested action. 7