IEEE P1003.2a Draft 8 – December 1991

5.37 `write` — Write to another user

5.37.1 Synopsis

`write` *user_name* [*terminal*]

5.37.2 Description

The `write` utility shall read lines from the user's standard input and write them to the terminal of another user. When first invoked, it shall write the message:

> `Message from`
> *sender-login-id*
> (*sending-terminal*)
> [*date*]...

to *user_name* . When it has successfully completed the connection, the sender's terminal shall be alerted twice to indicate that what the sender is typing is being written to the recipient's terminal.

If the recipient wants to reply, this can be accomplished by typing

> `write` *sender-login-id* [*sending-terminal*]

upon receipt of the initial message. Whenever a line of input as delimited by a 8
NL, EOF, or EOL special character (see POSIX.1 {8} section 7.1.1.9) is accumulated 8
while in canonical input mode, the accumulated data shall be written on the other 8
user's terminal. Characters shall be processed as follows: 8

— Typing the `<alert>` character shall write the `<alert>` character to the recipient's terminal.

— 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 cause `write` to write an appropriate message (`"EOT\n"` in the POSIX Locale) to the recipient's terminal and 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 written to the recipient's terminal.

To write to a user who is logged in more than once, the *terminal* argument can be used to indicate which terminal to write to; otherwise, the recipient's terminal is selected in an implementation-defined manner and an informational message shall be written to the sender's standard output, indicating which terminal was chosen.

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

5.37.3 Options

None.

5.37.4 Operands

The following operands shall be supported by the implementation:

user_name Login name of the person to whom the message shall be written. This operand shall be of the form returned by the `who` utility (see 5.36).

terminal Terminal identification in the same format provided by the `who` utility.

5.37.5 External Influences

5.37.5.1 Standard Input

Lines to be copied to the recipient's terminal shall be read from standard input. 7

5.37.5.2 Input Files

None.

5.37.5.3 Environment Variables

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

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 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 diagnostic messages should be written.

5.37.5.4 Asynchronous Events

If an interrupt signal is received, `write` shall write an appropriate message on the recipient's terminal and exit with a status of zero. It shall take the standard action for all other signals; see 2.11.5.4.

5.37.6 External Effects

5.37.6.1 Standard Output

An informational message shall be written to standard output if a recipient is logged in more than once.

5.37.6.2 Standard Error

Used only for diagnostic messages.

5.37.6.3 Output Files

The recipient's terminal shall be used for output.

5.37.7 Extended Description

None.

5.37.8 Exit Status

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

0 Successful completion.

>0 The addressed user is not logged on, or the addressed user denies permission.

5.37.9 Consequences of Errors

Default.

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

Usage, Examples

None.

History of Decisions Made

The `write` utility was included in the standard since it can be implemented on all terminal types. The working group considered the `talk` utility, which cannot be implemented on certain terminals, to be a ‘‘better’’ communications interface. Both of these programs are in widespread use on historical implementations. Therefore, the working group decided that both utilities should be specified.

The format of the terminal name is unspecified, but the descriptions of `ps`, `talk`, 7
`who`, and `write` require that they all use or accept the same format. 7

Annex E
(informative)
Rationale and Notes

This annex summarizes the deliberations of the IEEE P1003.2 working group, the committee charged by the IEEE Computer Society's Technical Committee on Operating Systems and Application Environments with devising an interface standard for a shell and related utilities to support and extend POSIX.1 {8}.

The annex is being published along with the standard to assist in the process of review. It contains historical information concerning the contents of the standard and why features were included or discarded by the working group. It also contains notes of interest to application programmers on recommended programming practices, emphasizing the consequences of some aspects of the standard that may not be immediately apparent.

Just as this standard relies on the knowledge of architecture, history, and definitions from the POSIX.1 {8}, so does this annex. The reader is referred to the Rationale and Notes appendix of POSIX.1 {8} for background material and bibliographic information about UNIX systems in general and POSIX specifically, which will not be duplicated here.

Editor's Note: The text of most of the Rationale has been temporarily located in Sections 1 through 5, adjacent to the text it is explaining. The text will return to this annex after the completion of balloting.

E.4 Excluded Utilities

Many utilities were evaluated by the developers of the standard; more utilities were excluded from the standard than included. The following list contains some common UNIX system utilities that were considered by the developers, but not included as part of the UPE or POSIX.2.

Editor's Note: This list will eventually be integrated with Annex E.4 in the base standard.

`SCCS` Both SCCS and RCS were originally considered as part of a version control utilities portion of the scope. However, this aspect of the UPE was abandoned by the working group. Their was broad consensus that SCCS was the most widely implemented system and that it had some technical advantages over RCS (other than