IEEE P1003.2a Draft 8 – December 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

3001 #### 5.13.5.3 Environment Variables

3002 The following environment variables shall affect the execution of `fg`:

3003 **LANG** This variable shall determine the locale to use for the
3004 locale categories when both **LC_ALL** and the correspond-
3005 ing environment variable (beginning with **LC_**) do not
3006 specify a locale. See 2.6.

3007 **LC_ALL** This variable shall determine the locale to be used to over-
3008 ride any values for locale categories specified by the set-
3009 tings of **LANG** or any environment variables beginning
3010 with **LC_**.

3011 **LC_CTYPE** This variable shall determine the locale for the interpreta-
3012 tion of sequences of bytes of text data as characters (e.g.,
3013 single- versus multibyte characters in arguments).

3014 **LC_MESSAGES** This variable shall determine the language in which mes-
3015 sages should be written.

3016 #### 5.13.5.4 Asynchronous Events

3017 Default.

3018 ### 5.13.6 External Effects

3019 #### 5.13.6.1 Standard Output

3020 The `fg` utility shall write the command line of the job to standard output in the
3021 following format:

3022 `"%s\n"`, *<command>*

3023 #### 5.13.6.2 Standard Error

3024 Used only for diagnostic messages.

3025 #### 5.13.6.3 Output Files

3026 None.

3027 ### 5.13.7 Extended Description

3028 None.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.13.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.13.9 Consequences of Errors

If job control is disabled, the `fg` utility shall exit with an error and no job shall be placed in the foreground.

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

Usage, Examples

The `fg` utility will not work as expected when it is operating in its own utility execution environment because that environment will have no applicable jobs to manipulate. See the rationale for `bg`. For this reason, `fg` is generally implemented as a shell regular built-in.

History of Decisions Made

The extensions to the shell specified in POSIX. 2 and POSIX. 2a have mostly been based on features provided by the KornShell. The job control features provided by `bg`, `fg`, and `jobs` are also based on the KornShell. The developers of the standard examined the characteristics of the C-Shell versions of these utilities and found that differences exist. Despite widespread use of the C-Shell, the KornShell versions were selected for this standard to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command-line editing features).

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.14 `file` — Determine file type

5.14.1 Synopsis

`file` *file* ...

5.14.2 Description

The `file` utility shall perform a series of tests on each specified *file* in an attempt to classify it:

(1) If the file is not a regular file, its file type shall be identified. The file types directory, FIFO , block special, and character special shall be identified as such. Other implementation-defined file types may also be identified.

(2) If the file is a regular file, and:

(a) The file is zero-length, it shall be identified as an empty file.

(b) The file is not zero-length, `file` shall examine an initial segment of the file and shall make a guess at identifying its contents or whether it is an executable binary file. (The answer is not guaranteed to be correct.)

If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.

5.14.3 Options

None.

5.14.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to be tested.

5.14.5 External Influences

5.14.5.1 Standard Input

None.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

3078 #### 5.14.5.2 Input Files

3079 The *file* can be any file type.

3080 #### 5.14.5.3 Environment Variables

3081 The following environment variables shall affect the execution of `file`:

3082 **LANG** This variable shall determine the locale to use for the
3083 locale categories when both **LC_ALL** and the correspond-
3084 ing environment variable (beginning with **LC_**) do not
3085 specify a locale. See 2.6.

3086 **LC_ALL** This variable shall determine the locale to be used to over-
3087 ride any values for locale categories specified by the set-
3088 tings of **LANG** or any environment variables beginning
3089 with **LC_**.

3090 **LC_CTYPE** This variable shall determine the interpretation of
3091 sequences of bytes of text data as characters (e.g., single-
3092 versus multibyte characters in arguments and input files).

3093 **LC_MESSAGES** This variable shall determine the language in which mes-
3094 sages should be written.

3095 #### 5.14.5.4 Asynchronous Events

3096 Default.

3097 ### 5.14.6 External Effects

3098 #### 5.14.6.1 Standard Output

3099 In the POSIX Locale, the following format shall be used to identify each operand,
3100 *file* specified:

3101 `"%s: %s\n"`, <*file*>, <*type*>

3102 The values for <*type*> are unspecified, except that in the POSIX Locale, if *file* is
3103 identified as one of the types listed in Table 5-100, <*type*> shall contain (but is not
3104 limited to) the corresponding string. Each space shown in the strings shall be
3105 exactly one <space> character.

3106 If the file named by the *file* operand does not exist, cannot be read, or the status of
3107 the file cannot be determined, the string `cannot open` shall be included as part
3108 of the <*type*> field, but this shall not be considered an error that affects the exit
3109 status.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Table 5-100 – `file` Output Strings

If *file* is a	<*type* > shall contain the string
directory	`directory`
FIFO	`fifo`
block special	`block special`
character special	`character special`
executable binary	`executable`
empty regular file	`empty`
`ar` archive library (see 6.1)	`archive`
extended `cpio` format (see POSIX .1 {8}10.1.2)	`cpio archive`
extended `tar` format (see POSIX .1 {8}10.1.1)	`tar archive`
shell script	`commands text`
C-language source	`c program text`
FORTRAN source	`fortran program text`

5.14.6.2 Standard Error

Used only for diagnostic messages.

5.14.6.3 Output Files

None.

5.14.7 Extended Description

None.

5.14.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.14.9 Consequences of Errors

Default.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

Usage, Examples

Historical systems have used a ‘‘magic file,’’ generally named `/etc/magic`, to help identify file types from their initial segments. Although the general format of the file is well known, it is not portable for more than a few file types; none of the interesting executable-guessing capabilities could be portable without a list of how all systems format their binary files. However, in the interest of some informal interoperability, the format is presented here for informative purposes only.

The file consists of four <blank>-separated fields:

offset A number specifying the offset, in bytes, into the file that is to be tested.

type The type of the data to be tested. The possible values for the type include `string`, `short`, `long`, and other data types from the C Standard {7} or other standards. Some types may be followed by a mask specifier of the form &*number* , which is AND ed with the value before any comparisons are done. The mask specifier is octal if preceded by a `0`, hexadecimal if preceded by a `0x`, and decimal otherwise.

value The value to be compared with the value from the file. If the value is a string, it may contain escape characters. If the value is numeric, it may be preceded by a character indicating the operation to be performed. Allowable values include:

- `=` The value from the file must equal the specified value.
- `<` The value from the file must be less than the specified value.
- `>` The value from the file must be greater than the specified value.
- `&` All bits in the specified value must be set in the value from the file.
- `^` At least one of the bits in the specified value must not be set in the value from the file.
- `x` Any value will match.

message The message to be printed if the comparison succeeds. If the string contains a *printf* () format specification, the value from the file is printed, using the message as the format string.

Some file formats contain additional information that is to be printed along with the file type. A line that begins with the character > indicates additional tests and messages to be printed. If the test on the line preceding the first line with a > succeeds, the test specified in all subsequent lines beginning with > are performed, and the messages printed if the tests succeed. The next line that does not begin with a > terminates this action.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Some example entries:

```
0     string   070707    ASCII cpio archive
0     string   !<arch>   portable archive
0     long     0550      executable
>12   long     >0        not stripped
```

History of Decisions Made

The `file` utility can only be required to ''guess'' at many of the file types because only exhaustive testing can determine some types with certitude. For example, binary data on some systems might match the initial segment of an executable or a `tar` archive. An exhaustive test might require that an executable be executed, or a C program be compiled, before certifying that the guess was correct. This is unreasonable. Since most historical systems generally provide very good guesses, `file` is in POSIX.2 with only the requirement that a best effort be made and that the output contain certain identifiable strings when the guess is made. Even on a very half-hearted system, the mandated output will be no less useful than that from `ls -l`.

The historical `-c` and `-m` options were omitted because the concept of a ''magic file'' is not portable.

The `-f` option was omitted because the same effect can (and should) be obtained using the `xargs` utility.

Historical versions of the `file` utility attempt to identify the following types of files: symbolic link, directory, character special, block special, socket, `tar` archive, `cpio` archive, SCCS archive, archive library, empty, `compress` output, `pack` output, binary data, C source, FORTRAN source, assembler source, `nroff/troff/eqn/tbl` source, `troff` output, shell script, C-shell script, English text, ASCII text, various executables, APL workspace, compiled terminfo entries, and CURSES screen images. Only those types that are reasonably well specified in POSIX or are directly related to POSIX.2 utilities are listed in the table.

Implementations that support symbolic links are encouraged to use the string `symbolic link` to identify them.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.15 jobs — Display status of jobs in the current session

5.15.1 Synopsis

```
jobs [ -l | -p ] [job_id ... ]
```

5.15.2 Description

The jobs utility shall display the status of jobs that were started in the current shell environment; see 3.12.

When jobs reports the termination status of a job, the shell shall remove its process ID from the list of those ''known in the current shell execution environment''; see 3.9.3.1. 7

5.15.3 Options

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

The following options shall be supported by the implementation:

-l (The letter ell.) Provide more information about each job listed. This information shall include the job number, current job, process group ID, state, and the command that formed the job.

-p Display only the process IDs for the process group leaders of the selected jobs.

By default, the jobs utility shall display the status of all stopped jobs, running background jobs, and all jobs whose status has changed and have not been reported by the shell. 7 7 7

5.15.4 Operands

The following operand shall be supported by the implementation:

job_id Specifies the jobs for which the status is to be displayed. If no *job_id* is given, the status information for all jobs shall be displayed. The format of *job_id* is described in 2.2.2.206.

5.15.5 External Influences

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

3235 #### 5.15.5.1 Standard Input

3236 None.

3237 #### 5.15.5.2 Input Files

3238 None.

3239 #### 5.15.5.3 Environment Variables

3240 The following environment variables shall affect the execution of `jobs`:

3241 **LANG** This variable shall determine the locale to use for the
3242 locale categories when both **LC_ALL** and the correspond-
3243 ing environment variable (beginning with **LC_**) do not
3244 specify a locale. See 2.6.

3245 **LC_ALL** This variable shall determine the locale to be used to over-
3246 ride any values for locale categories specified by the set-
3247 tings of **LANG** or any environment variables beginning
3248 with **LC_**.

3249 **LC_CTYPE** This variable shall determine the locale for the interpreta-
3250 tion of sequences of bytes of text data as characters (e.g.,
3251 single- versus multibyte characters in arguments).

3252 **LC_MESSAGES** This variable shall determine the language in which mes-
3253 sages should be written.

3254 #### 5.15.5.4 Asynchronous Events

3255 Default.

3256 ### 5.15.6 External Effects

3257 #### 5.15.6.1 Standard Output

3258 If the `-p` option is specified, the output shall consist of one line for each process
3259 ID:

3260 `"%d\n"`, *<process ID>*

3261 Otherwise, if the `-l` option is not specified, the output shall be a series of lines of
3262 the form:

3263 `"[%d] %c %s %s\n"`, *<job-number>* , *<current>* , *<state>*, *<command>*

3264 where the fields shall be as follows:

3265 *<current >* The character `+` identifies the job that would be used as a default
3266 for the `fg` or `bg` utilities; this job can also be specified using the
3267 *job_id* `"%+"` or `"%%"`. The character `-` identifies the job that

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

would become the default if the current default job were to exit; this job can also be specified using the *job_id* "%–". For other jobs, this field is a <space>. At most one job can be identified with + and at most one job can be identified with -. If there is any suspended job, then the current job shall be a suspended job. If there are at least two suspended jobs, then the previous job shall also be a suspended job.

<*job-number*>
A number that can be used to identify the process group to the `wait`, `fg`, `bg`, and `kill` utilities. Using these utilities, the job can be identified by prefixing the job number with %.

<*state*> One of the following strings (in the POSIX Locale):

`Running` Indicates that the job has not been suspended by a signal and has not exited.

`Done` Indicates that the job completed and returned exit status zero.

`Done(`*code*`)` Indicates that the job completed normally and that it exited with the specified nonzero exit status, *code*, expressed as a decimal number.

`Stopped`
`Stopped (SIGTSTP)`
Either indicates that the job was suspended by the SIGTSTP signal.

`Stopped (SIGSTOP)`
Indicates that the job was suspended by the SIGSTOP signal.

`Stopped (SIGTTIN)`
Indicates that the job was suspended by the SIGTTIN signal.

`Stopped (SIGTTOU)`
Indicates that the job was suspended by the SIGTTOU signal.

The implementation may substitute the string `Suspended` in place of `Stopped`. If the job was terminated by a signal, the format of <*state*> is unspecified, but it shall be visibly distinct from all of the other <*state*> formats shown here and shall indicate the name or description of the signal causing the termination.

<*command*>
The associated command that was given to the shell.

If the `-l` option is specified, a field containing the process group ID shall be inserted before the <*state*> field. Also, more processes in a process group may be output on separate lines, using only the process ID and <*command*> fields.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.15.6.2 Standard Error

Used only for diagnostic messages.

5.15.6.3 Output Files

None.

5.15.7 Extended Description

None.

5.15.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.15.9 Consequences of Errors

Default.

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

Usage, Examples

Both `%%` and `%+` are used to refer to the current job. Both forms are of equal validity—the `%%` mirroring `$$` and `%+` mirroring the output of `jobs`. Both forms reflect historical practice of the KornShell and the C-Shell with job control.

The `–p` option is historical practice in the KornShell and the System V `jsh`. It is the only way to find out the process group of a job because different implementations have different strategies for defining the process group of the job. Usage such as `$(jobs -p)` provides a way of referring to the process group of the job in an implementation-independent way.

The `jobs` utility will not work as expected when it is operating in its own utility execution environment because that environment will have no applicable jobs to manipulate. See the rationale for `bg`. For this reason, `jobs` is generally implemented as a shell regular built-in.

History of Decisions Made

The extensions to the shell specified in POSIX. 2 and POSIX. 2a have mostly been based on features provided by the KornShell. The job control features provided by `bg`, `fg`, and `jobs` are also based on the KornShell. The developers of the standard examined the characteristics of the C-Shell versions of these utilities and

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

found that differences exist. Despite widespread use of the C-Shell, the KornShell versions were selected for this standard to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command-line editing features).

The output for terminated jobs is left unspecified to accommodate various historical systems. The following formats have been witnessed:

(1) `Killed(`*signal name* `)`

(2) *signal name*

(3) *signal name* `(coredump)`

(4) *signal description* `- core dumped`

Most users should be able to figure these out, although it means that applications will have trouble parsing them.

The calculation of job IDs was not described since this would suggest an implementation, which may impose unnecessary restrictions.

In an earlier draft, a `–n` option was included to ‘‘Display the status of jobs that have changed, exited, or stopped since the last status report.’’ It was removed because the shell always writes any changed status of jobs before each prompt. Thus, `jobs –n` is a no-op.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.16 man — Display system documentation

5.16.1 Synopsis

```
man [-k] name ...
```

5.16.2 Description

The man utility shall write information about each of the *name* operands. If *name* is the name of a standard utility, man shall at a minimum write a message describing the syntax used by the standard utility, its options, and operands. If more information is available, the man utility shall provide it in an implementation-defined manner.

An implementation may provide information for values of *name* other than the standard utilities. Standard utilities that are listed as optional and that are not supported by the implementation either shall cause a brief message indicating that fact to be displayed or shall cause a full display of information as described previously.

5.16.3 Options

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

The following option shall be supported by the implementation:

-k Interpret *name* operands as keywords to be used in searching a utilities summary database that contains a brief ''purpose'' entry for each standard utility and write lines from the summary database that match any of the keywords. The keyword search shall produce results that are the equivalent of the output of the following command:

```
grep -Ei '
name
name
...
' summary-database
```

(This assumes that the *summary-database* is a text file with a single entry per line; this organization is not required and the example using grep -Ei is merely illustrative of the type of search intended.) The ''purpose'' entry to be included in the database shall consist of a terse description of the purpose of the utility.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.16.4 Operands

The following operand shall be supported by the implementation:

name A keyword or the name of a standard utility. When `-k` is not specified and *name* does not represent one of the standard utilities, the results are unspecified.

5.16.5 External Influences

5.16.5.1 Standard Input

None.

5.16.5.2 Input Files

None.

5.16.5.3 Environment Variables

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

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 in the summary database).

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

PAGER This variable shall be interpreted as a string representing an output filtering command for writing the output to a terminal. Any string acceptable as a *command_string* operand to the `sh -c` command (see 4.56), shall be valid. When standard output is a terminal device, the manual-page output shall be piped through the command. If the **PAGER** variable is null or not set, the command shall be either `more` (see 5.18) or another paginator utility documented in the system documentation.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

3429 #### 5.16.5.4 Asynchronous Events

3430 Default.

3431 ### 5.16.6 External Effects

3432 #### 5.16.6.1 Standard Output

3433 The `man` utility writes text describing the syntax of the utility *name,* its options,
3434 and its operands, or, when `-k` is specified, lines from the summary database. The
3435 format of this text is implementation defined.

3436 #### 5.16.6.2 Standard Error

3437 Used only for diagnostic messages.

3438 #### 5.16.6.3 Output Files

3439 None.

3440 ### 5.16.7 Extended Description

3441 None.

3442 ### 5.16.8 Exit Status

3443 The `man` utility shall exit with one of the following values:

3444 0 Successful completion.

3445 >0 An error occurred.

3446 ### 5.16.9 Consequences of Errors

3447 Default.

3448 ### 5.16.10 Rationale. *(This subclause is not a part of P1003.2a)*

3449 **Usage, Examples**

3450 The historical **MANPATH** variable is not included in POSIX because no attempt is
3451 made to specify naming conventions for manual page files, nor even to mandate
3452 that they are files at all. In some systems they could be a true database, or a
3453 hypertext file, or even fixed strings within the `man` executable. The developers of
3454 the standard consider the portability of manual pages to be outside their scope of
3455 work (and more appropriate to the working group developing application-

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

installation tools). However, users should be aware that **MANPATH** is implemented on a number of historical systems and that it can be used to tailor the search pattern for manual pages from the various categories (utilities, functions, file formats, etc.) when the system administrator reveals the location and conventions for manual pages on the system.

The keyword search can rely on at least the text of the clause titles from these utility descriptions and the implementation may add more keywords. The term ''clause titles'' refers to the strings such as

```
man - Display system documentation
ps - Report process status
```

History of Decisions Made

It is recognized that, as specified, the `man` utility is only of minimal usefulness. Opinion in the working group was strongly divided as to how much or how little information `man` should be required to provide. It was felt, however, that the provision of some portable way of accessing documentation would aid user portability. The arguments against a fuller specification were:

(1) Large quantities of documentation should not be required on a system cramped for disk space.

(2) The current manual system does not present information in a manner that greatly aids user-portability.

(3) A ''better help system'' is currently an area in which vendors feel that they can add value to their POSIX implementations.

The `-f` option was considered, but due to implementation differences, it was not included in the standard.

The description was changed to be more specific about what has to be displayed for a utility. The working group felt it was insufficient to allow a display of only the Synopsis, without giving a short description of what each option and operand does.

The ''purpose'' entry to be included in the database can be similar to the clause title (less the numeric prefix) from this standard for each utility. These titles are similar to those used in historical systems for this purpose.

See 4.40.10 for rationale concerning the default paginator.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.17 mesg — Permit or deny messages

5.17.1 Synopsis

```
mesg [ y | n ]
```

5.17.2 Description

The mesg utility shall control whether other users are allowed to send messages via write, talk, or other utilities to a terminal device. The terminal device affected shall be determined by searching for the first terminal in the sequence of devices associated with standard input, standard output, and standard error, respectively. With no arguments, mesg shall report the current state without changing it. Processes with appropriate privileges may be able to send messages to the terminal independent of the current state.

5.17.3 Options

None.

5.17.4 Operands

The following operands shall be supported by the implementation in the POSIX Locale:

y Grant permission to other users to send messages to the terminal device.

n Deny permission to other users to send messages to the terminal device.

5.17.5 External Influences

5.17.5.1 Standard Input

None.

5.17.5.2 Input Files

None.

5.17.5.3 Environment Variables

The following environment variables shall affect the execution of mesg:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 (from the `mesg` utility) are displayed.

5.17.5.4 Asynchronous Events

Default.

5.17.6 External Effects

5.17.6.1 Standard Output

If no operand is specified, `mesg` shall display the current terminal state in an unspecified format.

5.17.6.2 Standard Error

Used only for diagnostic messages.

5.17.6.3 Output Files

None.

5.17.7 Extended Description

None.

5.17.8 Exit Status

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

0 Receiving messages is allowed.

1 Receiving messages is not allowed.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

>1 An error occurred.

5.17.9 Consequences of Errors

Default.

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

Usage, Examples

The terminal changed by `mesg` is that associated with the standard input, output, or error, rather than the controlling terminal for the session. This is because users logged in more than once should be able to change any of their login terminals without having to stop the job running in those sessions. This is not a security problem involving the terminals of other users because appropriate privileges would be required to affect another user's terminals. 8 8

History of Decisions Made

The method of checking each of the first three file descriptors in sequence until a terminal is found was adopted from System V. 8 8

The file `/dev/tty` is not specified for the terminal device because it was thought to be too restrictive. Typical environment changes for the `n` operand are that write permissions are removed for *others* and *group* from the appropriate device. It was decided to leave the actual description of what is done as unspecified because of potential differences between implementations.

The format for standard output is unspecified because of differences between historical implementations. This output is generally not useful to shell scripts (they can use the exit status), so exact parsing of the output is unnecessary.

The mechanism by which the message status of the terminal is changed is unspecified. Therefore, unspecified actions may cause the status of the terminal to change after `mesg` has successfully completed. These actions may include, but are not limited to: another invocation of the `mesg` utility; login procedures; invocation of the `stty` utility; invocation of the `chmod` utility; etc.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.18 more — Display files on a page-by-page basis

5.18.1 Synopsis

more [−ceisu][−n *number*][−t *tagstring*][−p *command*][*file* ...] 8

Obsolescent Version:

more [−ceisu][−n *number*][+/*command*][−t *tagstring*][*file* ...] 8

5.18.2 Description

The more utility shall read files and either write them to the terminal on a page-by-page basis or filter them to standard output. If standard output is not a terminal device, all input files shall be copied to standard output in their entirety, 8 without modification. If standard output is a terminal device, the files shall be 8 written a number of lines (one ‘‘screenful’’) at a time under the control of user commands; see 5.18.7.

Certain block-mode terminals do not have all the capabilities necessary to support the complete more definition; they are incapable of accepting commands that are not terminated with a <newline>. Implementations that support such terminals shall provide an operating mode to more in which all commands can be terminated with a <newline> on those terminals. This mode shall:

— Be documented in the system documentation. 7

— At invocation, inform the user of the terminal deficiency that requires the <newline> usage and provide instructions on how this warning can be suppressed in future invocations.

— Not be required for implementations supporting only fully capable terminals.

— Not affect commands already requiring <newline>s.

— Not affect users on the capable terminals from using more as described in this standard.

5.18.3 Options

The more utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that +/*command* of the obsolescent version uses a nonstandard syntax.

The following options shall be supported by the implementation:

−c If a screen is to be written that has no lines in common with the current screen, or more is writing its first screen, more shall not scroll the screen, but instead shall redraw each line of the screen in turn, from the top of the screen to the bottom. In addition, if

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.