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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

3604 3605 3606 more is writing its first screen, the screen shall be cleared. This option may be ignored on devices that do not have the ability to clear to the end of a line or the end of a screen.

3607 3608 **−e** Exit immediately after writing the last line of the last file in the argument list; see 5.18.7.

3609 3610 **−i** Perform pattern matching in searches without regard to case. See 2.8.2.

3611 3612 3613 **−n** *number* Specify the number of lines per screenful. The *number* argument is a positive decimal integer. The **−n** option shall override any values obtained from the environment.

3614 **−p** *command* 8

3615 **+***command* (Obsolescent.) 8

3616 3617 3618 3619 3620 3621 For each file examined, initially execute the more command in the *command* argument. If the command is a positioning command, such as a line number or a regular expression search, set the current position (see 5.18.7) to represent the final results of the command, without writing any intermediate lines of the file. (For example, the two commands 8 8 8 8 8 8

```
more -p 1000j file
more -p 1000G file
```

3622 3623 8 8

3624 3625 3626 3627 3628 would be equivalent and start the display with the current position at line 1000, bypassing the lines that **j** would write and scroll off the screen if it had been issued during the file examination.) If the positioning command is unsuccessful, the first line in the file shall be the current position. 8 8 8 8 8

3629 **−s** Replace consecutive empty lines with a single empty line.

3630 **−t** *tagstring*

3631 3632 3633 3634 3635 3636 Write the screenful of the file containing the tag named by the *tagstring* argument. See the ctags utility in 5.7. The tags feature represented by **−t** *tagstring* and the **:t** command (see 5.18.7.23) is optional. It shall be provided on any system that also provides a conforming implementation of ctags; otherwise, the use of **−t** produces undefined results. 8 8 7

3637 3638 3639 3640 3641 3642 **−u** Treat <backspace> as a printable control character, displayed as an implementation-defined character sequence (see 5.18.7), suppressing backspacing and the special handling that produces underlined or standout-mode text on some terminal types. Also, do not ignore a <carriage-return> character at the end of a line. 7 7 7 7 7 7

3643 3644 3645 If both the **−t** *tagstring* and **−c** *command* (or the obsolescent **+/***command*) options are given, the **−t** *tagstring* shall be processed first; i.e., the file containing the tag is selected by **−t** and then the command is executed. 8 8 8

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

5.18.4 Operands

The following operand shall be supported by the implementation:

file A pathname of an input file. If no *file* operands are specified, the standard input shall be used. If a *file* is –, the standard input shall be read at that point in the sequence.

5.18.5 External Influences

5.18.5.1 Standard Input

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

5.18.5.2 Input Files

The input files being examined shall be text files. If standard output is a terminal, standard error shall be used to read commands from the user. If standard output is a terminal, standard error is not readable, and command input is needed, `more` shall terminate with an error indicating that it was unable to read user commands. If standard output is not a terminal, no error shall result if standard error cannot be opened for reading.

5.18.5.3 Environment Variables

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

COLUMNS This variable shall override the system-selected horizontal screen size. See 2.6 for valid values and results when it is unset or null.

EDITOR This variable shall be used by the `v` command to select an editor; see 5.18.7.

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_COLLATE This variable shall determine the locale for character collation information in regular expressions.

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

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), and the behavior of character classes within regular expressions.

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

LINES This variable shall override the system-selected vertical screen size, used as the number of lines in a screenful. See 2.6 for valid values and results when it is unset or null. The `–n` option shall take precedence over the **LINES** variable for determining the number of lines in a screenful.

MORE This variable shall be interpreted as a string containing options described in 5.18.3, preceded with hyphens and `<blank>`-separated as on the command line. Any command-line options shall be processed after those in the **MORE** variable, as if the command line were:

```
more $MORE options operands
```

The **MORE** variable shall take precedence over the **TERM** and **LINES** variables for determining the number of lines in a screenful.

TERM This variable shall be interpreted as the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.

5.18.5.4 Asynchronous Events

Default.

5.18.6 External Effects

5.18.6.1 Standard Output

The standard output shall be used to write the contents of the input files.

5.18.6.2 Standard Error

Used for diagnostic messages and user commands (see 5.18.5.2), and, if standard output is a terminal device, to write a prompting string. The prompting string shall only appear as the last line of the ''screen'' and shall contain the name of the file currently being examined, but it is otherwise unspecified. User input for the `/`, `?`, `:e`, and `:t` commands shall be written on the same line of the screen as the prompt. It is unspecified if informational messages are written for other user

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

commands.

5.18.6.3 Output Files

None.

5.18.7 Extended Description

The number of lines available per ‘‘screen’’ shall be determined by the –n option, if present, or by examining values in the environment (see 5.18.5.3). If neither method yields a number, an unspecified number of lines shall be used. The actual number of lines written shall be one less than this number, as the last line of the screen shall be used to write a user prompt and user input. If the number of lines available per screen is less than four, the results are undefined.

In the following descriptions, the ‘‘current position’’ refers to two things:

— The position of the current line on the screen.

— The line number (in the file) of the current line on the screen.

Usually, the line on the screen corresponding to the current position is the third line on the screen. If this is not possible (there are fewer than three lines to display, or this is the first page of the file, or it is the last page of the file), then the current position is either the first or last line on the screen as described later.

The number of columns available per line shall be determined by examining values in the environment (see 5.18.5.3), with a default value as described in 2.6. The `more` utility shall write lines containing more characters than would fit into this number of columns by breaking the line into one or more ‘‘logical’’ lines where each of these lines but the last contains the number of characters needed to fill the columns. The logical lines shall be written independently of each other; i.e., commands affecting a single line shall affect them separately.

When standard output is a terminal and –u is not specified, `more` shall treat <backspace>s and <carriage-return>s specially:

— A character, followed first by a <backspace>, then by an underscore (_), shall cause that character to be written as underlined text, if the terminal type supports that. An underscore, followed first by a <backspace>, then any character, shall also cause that character to be written as underlined text, if the terminal type supports that.

— A <backspace> that appears between two identical printable characters shall cause the first of those two characters to be written as emboldened text (i.e., visually brighter, standout mode, or inverse-video mode), if the terminal type supports that, and the second to be discarded. Immediately subsequent occurrences of backspaces/character pairs for that same character shall also be discarded. (For example, the sequence `a\ba\ba\ba` is interpreted as a single emboldened `a`.)

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

— Other <backspace> sequences shall be written directly to the terminal, which generally cause the character preceding the <backspace> to be suppressed in the display.

— A <carriage-return> at the end of a line shall be ignored, rather than being written as a control character, as described in the next paragraph.

It is implementation defined how other nonprintable characters are written. Implementations should use the same format that they use for the ex print command; see 5.10.7.2.21.

If the −t option was specified, more shall write a screen of the file containing the specified tag in the current position.

If the −p option was specified, more shall find the first occurrence of a line in the input file that contains characters matching the specified pattern and write the screen so that the line found is in the current position.

If neither the −p or −t options were specified, more shall write the first screen of the first input file.

Once the initial screen has been written, more shall prompt the user and, based on the interactive user input, shall modify the screen or exit. If the modification of the screen results in a screen that has lines in common with the current screen, more shall scroll the screen rather than clearing and redrawing the screen (unless the −c option is specified). If the modification of the screen results in a screen that has no lines in common with the current screen, more shall clear and redraw the screen.

If the standard output is not a terminal device, more shall always exit when it reaches end-of-file on the last file in its argument list. Otherwise, for all files but the last, more shall prompt, with an indication that it has reached the end of file, along with the name of the next file. For the last file specified, or for the standard input if no file is specified, more shall prompt, indicating end-of-file, and accept additional commands. If the next command specifies forward scrolling, more shall exit. If the −e option is specified, more shall exit immediately after writing the last line of the last file.

Several of the commands described in this clause move the current position backwards in the input stream. In the case that text is being taken from a nonrewindable stream, such as a pipe, it is implementation defined how much backward motion is supported.

If a scrolling command cannot be performed because there are insufficient lines to scroll, more shall alert the terminal. The scrolling commands are b, control-B, d, control-D, f, control-F, g, G, j, k, s, u, and control-U.

The interactive commands in the following subclauses shall be supported. Some commands can be preceded by a decimal integer, called *count* in the following descriptions. If not specified with the command, *count* shall default to 1.

In the following descriptions, *pattern* is a basic regular expression, as described in 2.8.3. The term ‘‘examine’’ is historical usage meaning ‘‘open the file for viewing’’; for example, more foo would be expressed as ‘‘examining’’ file foo.

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

3797 #### 5.18.7.1 Help

3798 *Synopsis* : `h`

3799 Write a summary of these commands and other implementation-defined com-
3800 mands.

3801 #### 5.18.7.2 Move forward one screenful

3802 7

3803 *Synopsis* : [*count*]`f`
3804 *Synopsis* : [*count*]`control-F`

3805 Move forward *count* lines, with a default of one screenful. At end-of-file, `more` 7
3806 shall continue with the next file in the list, or exit if the current file is the last file
3807 in the list.

3808 #### 5.18.7.3 Move backward one screenful

3809 *Synopsis* : [*count*]`b`
3810 *Synopsis* : [*count*]`control-B`

3811 Move backward *count* lines, with a default of one screenful (see option `–n`). If
3812 *count* is more than the screen size, only the final screenful shall be written.

3813 #### 5.18.7.4 Scroll forward one line

3814 *Synopsis* : [*count*]`<space>` 7
3815 *Synopsis* : [*count*]`j`
3816 *Synopsis* : [*count*]`<newline>`

3817 Scroll forward *count* lines. The default *count* for `<space>` shall be one screenful; 7
3818 for `j` and `<newline>`, one line. The entire *count* lines shall be written, even if 7
3819 *count* is more than the screen size. At end-of-file, `<newline>` shall cause `more` to
3820 continue with the next file in the list, or exit if the current file is the last file in
3821 the list.

3822 #### 5.18.7.5 Scroll backward one line

3823 *Synopsis* : [*count*]`k`

3824 Scroll backward *count* lines, with a default of 1. The entire *count* lines shall be
3825 written, even if *count* is more than the screen size.

3826 #### 5.18.7.6 Scroll forward one-half screenful

3827 *Synopsis* : [*count*]`d`
3828 *Synopsis* : [*count*]`control-D`

3829 Scroll forward *count* lines, with a default of one half of the screen size. If *count* is
3830 specified, it shall become the new default for subsequent `d` and `u` commands.

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

3831 #### 5.18.7.7 Skip forward one line

3832 *Synopsis* : [*count*]s

3833 Skip forward *count* line(s), with a default of 1, and write the next screenful begin-
3834 ning at that point. If *count* would cause the current position to be such that less
3835 than one screenful would be written, the last screenful in the file shall be written.

3836 #### 5.18.7.8 Scroll backward one-half screenful

3837 *Synopsis* : [*count*]u
3838 *Synopsis* : [*count*]control-U

3839 Scroll backward *count* lines, with a default of one half of the screen size. If *count*
3840 is specified, it shall become the new default for subsequent d and u commands.

3841 #### 5.18.7.9 Go to beginning of file

3842 *Synopsis* : [*count*]g

3843 Go to line *count* in the file, with a default of 1 (beginning of file). Scroll or rewrite 7
3844 the screen so that that line is at the current position. 7

3845 #### 5.18.7.10 Go to end of file

3846 *Synopsis* : [*count*]G

3847 Go to line *count* in the file, with a default of the end of the file. If no *count* is 8
3848 specified, scroll or rewrite the screen so that the last line in the file is at the bot- 8
3849 tom of the screen. If *count* is specified, scroll or rewrite the screen so that that 8
3850 line is at the current position. 8

3851 #### 5.18.7.11 Refresh the screen

3852 *Synopsis* : r
3853 *Synopsis* : control-L

3854 Refresh the screen.

3855 #### 5.18.7.12 Discard and refresh

3856 *Synopsis* : R

3857 Refresh the screen, discarding any buffered input. If the current file is nonseek- 8
3858 able, buffered input shall not be discarded and the R command is equivalent to 8
3859 the r command. 8

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

5.18.7.13 Mark position

Synopsis : `m`*letter*

Mark the current position with the letter named by *letter* , where *letter* represents the name of one of the lowercase letters of the portable character set. When a new file is examined, all marks may be lost.

5.18.7.14 Return to mark

Synopsis : `'`*letter*

Return to the position that was previously marked with the letter named by *letter* , making that line the current position.

5.18.7.15 Return to previous position

Synopsis : `''`

Return to the position from which the last large movement command was executed (where a ‘‘large movement’’ is defined as any movement of more than a screenful of lines). If no such movements have been made, return to the beginning of the file.

5.18.7.16 Search forward for pattern

Synopsis : [*count*]`/`[`!`]*pattern* `<newline>`

Search forward in the file for the *count* -th line containing the *pattern* . The *count* shall default to 1. The search shall start at the line following the current position. If the search is successful, the screen shall be modified so that the searched-for line is in the current position. The null regular expression (`/<newline>`) shall repeat the search using the previous regular expression. If the character `!` is included, the line for searching shall be those that do not contain the *pattern* .

If a match is found for the *pattern* , the logical line containing the pattern shall be written in the current position. If the matching line is already on the screen, the screen shall be scrolled to make that line the current position; otherwise, a full screen shall be written. If no match is found for the pattern, a message to that effect shall be written as a part of the prompt, and the screen and the current position shall remain unchanged. However, if no match is found and the input is the standard input, the screen may be scrolled to the last screenful of the input.

5.18.7.17 Search backward for pattern

Synopsis : [*count*]`?`[`!`]*pattern* `<newline>`

Search backward in the file for the *count* -th line containing the *pattern* . The search shall start at the line immediately before the current position. If the search is successful, the screen shall be modified so that the searched-for line is in the current position. The null regular expression (`?<newline>`) shall repeat the

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

search using the previous regular expression. If the character ! is included, the line for searching shall be those that do not contain the *pattern* .

If a match is found for the *pattern* , the logical line containing the pattern shall be written in the current position. If the matching line is already on the screen, the screen shall be scrolled to make that line the current position; otherwise, a full screen shall be written. If no match is found for the pattern, a message to that effect shall be written as a part of the prompt, and the screen and the current position shall remain unchanged. However, if no match is found and the input is the standard input, the screen may be scrolled to the last screenful of the input.

5.18.7.18 Repeat search

Synopsis : [*count*]n

Repeat the previous search for *count* -th line (default 1) containing the last *pattern* (or not containing the last *pattern* , if the previous search was /! or ?!).

5.18.7.19 Repeat search in reverse

Synopsis : [*count*]N

Repeat the search in the opposite direction of the previous search for the *count* -th line (default 1) containing the last *pattern* (or not containing the last *pattern* , if the previous search was /! or ?!).

5.18.7.20 Examine new file

Synopsis : :e [*filename*]<newline>

Examine a new file. If the *filename* argument is not specified, the ‘‘current’’ file (see the :n and :p commands below) from the list of files in the command line shall be re-examined. The *filename* shall be subjected to the process of shell word expansions (see 3.6); if more than a single pathname results, the effects are unspecified. If *filename* is a number sign (#), the previously examined file shall be re-examined. If *filename* refers to a nonseekable file, the results are unspecified.

5.18.7.21 Examine next file

Synopsis : [*count*]:n

Examine the next file. If a number *count* is specified, the *count* -th next file shall be examined. If *filename* refers to a nonseekable file, the results are unspecified.

5.18.7.22 Examine previous file

Synopsis : [*count*]:p

Examine the previous file. If a number *count* is specified, the *count* -th previous file shall be examined. If *filename* refers to a nonseekable file, the results are unspecified.

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

3931 #### 5.18.7.23 Go to tag

3932 *Synopsis* : `:t` *tagstring* `<newline>`

3933 Go to the supplied *tagstring* and scroll/rewrite the screen with that line in the 7
3934 current position; see the `−t` option. If the `ctags` utility is not supported by the 8
3935 system, the use of `:t` produces undefined results.

3936 #### 5.18.7.24 Invoke editor

3937 *Synopsis* : `v`

3938 Invoke an editor to edit the current file being examined. If standard input is
3939 being examined, the results are unspecified. The name of the editor shall be
3940 taken from the environment variable **EDITOR** , or shall default to `vi`. If **EDITOR** 7
3941 represents either `vi` or `ex`, the editor shall be invoked with options such that the 7
3942 current editor line is the physical line corresponding to the current position in 7
3943 `more` at the time of invocation. For example, either `ex` or `vi` is invoked by speci- 7
3944 fying the editor name and following that with `−c` *linenumber* . It is implementa- 7
3945 tion defined whether line-setting options are passed to editors other than `vi` and 7
3946 `ex`. 7

3947 The file types that can be edited are implementation defined. 7

3948 When the editor exits, `more` shall resume on the current file by rewriting the 7
3949 screen with the current line as the current position. 7

3950 #### 5.18.7.25 Display position

3951 *Synopsis* : `=`
3952 *Synopsis* : `control-G`

3953 Write the name of the file currently being examined, the number relative to the
3954 total number of files there are to examine, the current line number, the current
3955 byte number and the total bytes to write, and what percentage of the file precedes
3956 the current position. If `more` is reading from standard input, or the file is shorter
3957 than a single screen, some of these items need not be written. All of these items
3958 shall reference the first byte of the line after the last line written.

3959 #### 5.18.7.26 Quit

3960 *Synopsis* : `q`
3961 *Synopsis* : `:q`
3962 *Synopsis* : `ZZ`

3963 Exit `more`.

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

5.18.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.18.9 Consequences of Errors

If an error is encountered accessing a file when using the `:n` command, `more` shall attempt to examine the next file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the `:p` command, `more` shall attempt to examine the previous file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the `:e` command, `more` shall remain in the current file and the final exit status shall not be affected.

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

Usage, Examples

The text stating ''from a nonrewindable stream ... implementations may limit the amount of backward motion supported'' would allow an implementation that permitted no backward motion beyond text already on the screen. It was not possible to require a minimum amount of backward motion that would be effective for all conceivable device types. The implementation should allow the user to back up as far as possible, within device and reasonable memory allocation constraints.

The `-p` allows arbitrary commands to be executed at the start of each file. Examples are:

`more -p G file1 file2` Examine each file starting with its last screenful.

`more -p 100 file1 file2` Examine each file starting with line 100 in the current position (usually the third line, so line 98 would be the first line written).

`more -p /100 file1 file2` Examine each file starting with the first line containing the string `100` in the current position

History of Decisions Made

The `more` utility, available in BSD and BSD-derived systems, was chosen as the prototype for the POSIX.2 file display program since it is more widely available than either the public-domain program `less` or `pg`, a pager provided in System V. The 4.4 BSD `more` is the model for the features selected; it is almost fully upward-compatible from the 4.3 BSD version in wide use and has become more amenable for `vi` users. Several features originally derived from various file editors, found

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

in both less and pg, have been added to this specification as they have proved extremely popular with users.

There are inconsistencies between more and vi that result from historical practice. For example, the single-character commands h, f, b, and <space> are screen movers in more, but cursor movers in vi. These inconsistencies were maintained because the cursor movements are not applicable to more and the powerful functionality achieved without the use of the control key justifies the differences in most users' minds.

The tags interface has been included in a program that is not a text editor because it promotes another degree of consistent operation with vi. It is conceivable that the paging environment of more would be superior for browsing source code files in some circumstances.

The operating mode referred to for block-mode terminals effectively adds a <newline> to each synopsis line that currently has none. So, for example, d<newline> would page one screenful. The mode could be triggered by a command-line option, environment variable, or some other method. The details are not imposed by POSIX. 2 because there are so few systems known to support such terminals. Nevertheless, it was felt that all systems should be able to support more given the exception cited for this small community of terminals because, in comparison to vi, the cursor movements are few and the command set relatively amenable to the optional <newline>s.

Some versions of more provide a shell escaping mechanism similar to the ex ! command. The working group did not feel this was necessary in a paginator, particularly given the wide acceptance of multiple window terminals and job control features. (It chose to retain such features in the editors and mailx because the shell interaction also gives an opportunity to modify the editing buffer, which is not applicable to more).

The −p (position) option replaces the + command because of the utility syntax guidelines. In earlier drafts, it took a *pattern* argument, but historical less provided the more general facility of a command. It would have been desirable to use the same −c as ex and vi, but the letter was already in use.

When the standard output is not a terminal, none of the filter-modification options are effective. This is based on historical practice. For example, a typical implementation of man pipes its output through more −s to squeeze excess white space for terminal users. When man is piped to lp, however, it is undesirable for this squeezing to happen.

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

5.19 newgrp — Change to a new group

5.19.1 Synopsis

```
newgrp [-l][group]
```

Obsolescent Version:

```
newgrp [-][group]
```

5.19.2 Description

The newgrp utility shall create a new shell execution environment with a new real and effective group identification. Of the attributes listed in 3.12, the new shell execution environment shall retain the working directory, file creation mask, and exported variables from the previous environment (i.e., open files, traps, unexported variables, alias definitions, shell functions, and set options may be lost). All other aspects of the process environment that are preserved by the *exec* family of functions in POSIX.1 {8} also shall be preserved by newgrp; whether other aspects are preserved is unspecified.

A new shell execution environment shall be created whether or not newgrp is able to change to the requested new group.

If the POSIX.1 {8} {NGROUPS_MAX} is greater than 1, newgrp shall add the specified group to the supplemental groups for the process, if there is room to add another group.

With no operands, newgrp shall change the effective group back to the group(s) identified in the user's user entry, and if {NGROUPS_MAX} is greater than 1, shall set the list of supplementary groups to that set in the user's group database entries.

If a password is required for the specified group, and the user is not listed as a member of that group in the group database, the user shall be prompted to enter the correct password for that group. If the user is listed as a member of that group, no password shall be requested. If no password is required for the specified group, it is implementation defined whether users not listed as members of that group can change to that group. Whether or not a password is required, implementation-defined system accounting mechanisms may impose additional authorization restrictions that may cause newgrp to exit with an error status.

5.19.3 Options

The newgrp utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the obsolescent version uses – in a nonstandard manner.

The following option shall be supported by the implementation:

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

`-l` (The letter ell.)

`-` (Obsolescent.)

Change the environment to what would be expected if the user actually logged in again.

5.19.4 Operands

The following operand shall be supported by the implementation:

group A group name from the group database or a nonnegative numeric group ID. Either specifies the group ID to which the real and effective group IDs shall be set. If *group* is a nonnegative numeric string and exists in the group database as a group name [see *getgrnam* () in POSIX.1 {8} 9.2.1], the numeric group ID associated with that group name shall be used as the group ID.

5.19.5 External Influences

5.19.5.1 Standard Input

None.

5.19.5.2 Input Files

The file `/dev/tty` shall be used to read a single line of text for password checking, when one is required.

5.19.5.3 Environment Variables

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

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).

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

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

5.19.5.4 Asynchronous Events

Default.

5.19.6 External Effects

5.19.6.1 Standard Output

None.

5.19.6.2 Standard Error

Used for diagnostic messages and a prompt string for a password, if one is required.

5.19.6.3 Output Files

None.

5.19.7 Extended Description

None.

5.19.8 Exit Status

If `newgrp` succeeds, the exit status shall be the exit status of the shell. Otherwise, the `newgrp` utility shall exit with the following value:

>0 An error occurred.

5.19.9 Consequences of Errors

The invoking shell may terminate.

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

Usage, Examples

None.

History of Decisions Made

Most historical implementations use one of the *exec* functions to implement the behavior of `newgrp`. Errors detected before the *exec* leave the environment unchanged, while errors detected after the *exec* leave the user in a changed environment. While it would be useful to have `newgrp` issue a diagnostic

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

message to tell the user that the environment changed, it would be inappropriate to require this change to some historical implementations.

The password mechanism is allowed in the group file, but how this would be implemented is not specified.

The `newgrp` utility was retained in POSIX.2 even given the existence of the multiple group permissions feature in POSIX.1 {8}, because it allows files to be created with a specific group ownership. In some systems, the group ownership is determined by the group of the parent directory in which a file is created, as allowed by POSIX.1 {8}; on such systems, `newgrp` does not affect the group ownership in directories where the parent directory dictates it. However, POSIX.1 {8} allows systems to omit this feature; therefore, the `newgrp` functionality is still required on those systems.

5.20 `nice` — Invoke a utility with an altered system scheduling priority

5.20.1 Synopsis

`nice [-n` *increment* `]` *utility* `[`*argument* `...]`

Obsolescent Version:

`nice [-`*increment* `]` *utility* `[`*argument* `...]`

5.20.2 Description

The `nice` utility shall invoke a utility, requesting that it be run with a different system scheduling priority (see 2.2.2.212). With no options and only if the user has appropriate privileges, the executed utility shall be run with a system scheduling priority that is some implementation-defined quantity less than or equal to the system scheduling priority of the current process. If the user lacks appropriate privileges to affect the system scheduling priority in the requested manner, the `nice` utility shall not affect the system scheduling priority; in this case, a warning message may be written to standard error, but this shall not prevent the invocation of *utility* or affect the exit status.

5.20.3 Options

The `nice` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the obsolescent version allows a multidigit decimal integer as an option name.

The following option shall be supported by the implementation:

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

4165 –n *increment*
4166 -*increment* (Obsolescent.)
4167 Specify how the system scheduling priority of the executed utility
4168 shall be adjusted. The *increment* option-argument shall be a posi-
4169 tive or negative decimal integer that shall be used to modify the
4170 system scheduling priority of the executed utility in an
4171 implementation-defined manner.

4172 Positive *increment* values shall cause a lower or unchanged sys-
4173 tem scheduling priority. Negative *increment* values may require
4174 appropriate privileges and shall cause a higher or unchanged sys-
4175 tem scheduling priority.

4176 The system scheduling priority shall be bounded in an
4177 implementation-defined manner. If the requested *increment*
4178 would raise or lower the system scheduling priority of the exe-
4179 cuted utility beyond implementation-defined limits, then the limit
4180 whose value was exceeded shall be used.

4181 ### 5.20.4 Operands

4182 The following operands shall be supported by the implementation:

4183 *utility* The name of a utility that is to be invoked. If the *utility* operand
4184 names any of the special built-in utilities in 3.14, the results are
4185 undefined.

4186 *argument* Any string to be supplied as an argument when invoking the util-
4187 ity named by the *utility* operand.

4188 ### 5.20.5 External Influences

4189 #### 5.20.5.1 Standard Input

4190 None.

4191 #### 5.20.5.2 Input Files

4192 None.

4193 #### 5.20.5.3 Environment Variables

4194 The following environment variables shall affect the execution of `nice`:

4195 **LANG** This variable shall determine the locale to use for the
4196 locale categories when both **LC_ALL** and the correspond-
4197 ing environment variable (beginning with **LC_**) do not
4198 specify a locale. See 2.6.

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

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.

PATH This variable shall determine the search path used to locate the utility to be invoked. See 2.6.

5.20.5.4 Asynchronous Events

Default.

5.20.6 External Effects

5.20.6.1 Standard Output

None.

5.20.6.2 Standard Error

Used only for diagnostic messages.

5.20.6.3 Output Files

None.

5.20.7 Extended Description

None.

5.20.8 Exit Status

If the *utility* utility is invoked, the exit status of `nice` shall be the exit status of *utility;* otherwise, the `nice` utility shall exit with one of the following values:

1–125 An error occurred in the `nice` utility.

126 The utility specified by *utility* was found but could not be invoked.

127 The utility specified by *utility* could not be found.

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

5.20.9 Consequences of Errors

Default.

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

Usage, Examples

Note that, in the obsolescent version, `–5` is a positive *increment*, while `––5` is a negative *increment*. While this might appear counterintuitive, it is how historical versions of `nice` work.

The only guaranteed portable uses of this utility are:

`nice` *utility* Run *utility* with the default lower system scheduling priority.

`nice -n` ‹*positive integer*› *utility*
Run *utility* with a lower system scheduling priority.

On some systems they will have no discernible effect on the invoked utility and on others they will be exactly equivalent.

Historical systems have frequently supported the ‹*positive integer*› up to 20. Since there is no error penalty associated with guessing a number that is too high, users without access to the system conformance document (to see what limits are actually in place) could use the historical 1 to 20 range or attempt to use very large numbers if the job should be truly low priority.

The system scheduling priority value of a process can be displayed using the command:

```
ps -o nice
```

The `command`, `env`, `nice`, `nohup`, `time`, and `xargs` utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish ‘‘failure to find a utility’’ from ‘‘invoked utility exited with an error indication.’’ The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for ‘‘normal error conditions’’ and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for any other reason. In System V, the exit status from `nice`, when `nice` detects an error, is 2. In 4.3 BSD the status is 1.

History of Decisions Made

Due to the text about the limits of the system scheduling priority being implementation defined, `nice` is not required to actually change the priority of the executed command; the limits could be zero differences from the system default, although

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

the implementor is required to document this fact in the conformance document. Because of this, including this utility in POSIX.2 does not require implementations to support something like the traditional *nice*() system call. In particular, a POSIX.2 implementation could run on a nonextended POSIX.1 {8} implementation simply by supplying the following `nice` utility. (The example supports only the nonobsolescent form for illustrative purposes.)

```
#include <stdio.h>
extern int      optind;
extern char     *optarg;
#define USAGE   "Usage: nice [-n incr] <utility> [<arg> ...]\n"

int main(int argc, char *argv[]);
{
        int     i;

        while ((i = getopt(argc, argv, "n:")) != EOF) {
                switch (i) {
                    default:
                        fprintf(stderr, USAGE);
                        exit(127);
                    case 'n':  /* ignore increment */
                        break;
                }
        }
        argc -= optind;
        argv += optind;
        if (argc > 0) {
                execvp(argv[0], &argv[0]);
                perror(argv[0]);
        } else {
                fprintf(stderr, USAGE);
        }
        exit(127);
}
```

On systems that choose this approach, `nice` is a benign no-op. On the vast majority of systems that support `nice` historically, it becomes more useful. The developers of the standard considered this a quality of implementation issue and resisted calls to omit `nice` unless it could be guaranteed to have a discernible effect on all systems. The efficacy of `nice` is a legitimate procurement consideration.

The 4.3 BSD version of `nice` does not check if *increment* is a valid decimal integer. The command `"nice -x utility"`, for example, would be treated the same as the command `"nice --1 utility"`. If the user does not have appropriate privileges, this will result in a ‘‘permission denied’’ error. This is considered a bug.

When a user without appropriate privileges gives a negative *increment*, System V treats it like the command `"nice -0 utility"`, while 4.3 BSD writes a ‘‘permission denied’’ message and does not run the utility. Neither was considered clearly superior, so the behavior was left unspecified.

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