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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

4.56.7.4 Vi Line Editing Insert Mode

While in insert mode, any character typed shall be inserted in the current command line, unless it is from the following set.

<newline> Execute the current command line being edited.

erase Delete the character previous to the current cursor position and move the current cursor position back one character. In insert mode, characters shall be erased from both the screen and the buffer when backspacing.

interrupt Terminate command-line editing with the same effects as described for interrupting command mode; see 4.56.7.3.

kill Clear all the characters from the input line.

control-V Insert the next character input, even if the character is otherwise a special insert mode character.

control-W Delete the characters from the one preceding the cursor to the preceding word boundary. The word boundary in this case is the closer to the cursor of either the beginning of the line or a character that is in neither the `blank` nor `punct` character classification of the current locale.

\ If followed by the *erase* or *kill* character, that character shall be inserted into the input line. Otherwise, the backslash itself shall be inserted into the input line.

end-of-file Interpreted as the end of input in `sh`. This interpretation shall occur only at the beginning of an input line. If end-of-file is entered other than at the beginning of the line, the results are unspecified.

<ESC> Place `sh` into command mode.

4.56.7.5 Vi Line Editing Command Mode

In command mode for the command-line editing feature, decimal digits not beginning with 0 that precede a command letter shall be remembered. Some commands use these decimal digits as a count number that affects the operation.

The term *motion command* in the following list represents one of the commands:

```
<space> 0       b       F       l       W       ^
$       ;       E       f       T       w       |
,       B       e       h       t
```

Any command that modifies the current line shall cause a copy of the current line to be made at the end of the command history, the current line shall become that copy, and the edit shall be performed on that copy.

Any command that is preceded by *count* shall take a count (the numeric value of any preceding decimal digits). Unless otherwise noted, this count shall cause the

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

specified operation to repeat by the number of times specified by the count. Also unless otherwise noted, a *count* that is out of range is considered an error condition and shall alert the terminal, but neither the cursor position, nor the command line, shall change.

The terms *word* and *bigword* are used as defined in the `vi` description; see 5.35.7.1. The term *save buffer* corresponds to the term *unnamed buffer* in 5.35.7.1. 7 7

The following commands shall be recognized in command mode:

`<newline>` Execute the current command line being edited.

`control-L` Redraw the current command line. Position the cursor at the same location on the new command line.

`#` Insert the character # at the beginning of the current command line and treat the current command line as a comment. This line shall be entered into the command history; see 5.12. 8 8

`=` Display the possible shell word expansions (see 3.6) of the bigword at the current command-line position. These expansions shall be displayed on subsequent terminal lines. If the bigword contains none of the characters `?`, `*`, or `[`, an asterisk (`*`) shall be implicitly assumed at the end. If any directories are matched, these expansions shall have a `/` character appended. After the expansion, the line shall be redrawn, the cursor shall be repositioned at the current cursor position, and `sh` shall be placed in command mode.

`\` Perform pathname expansion (see 3.6.6) on the current bigword, up to the largest set of characters that can be matched uniquely. If the bigword contains none of the characters `?`, `*`, or `[`, an asterisk (`*`) shall be implicitly assumed at the end. This maximal expansion then shall replace the original bigword in the command line, and the cursor shall be placed after this expansion. If the resulting bigword completely and uniquely matches a directory, a `/` character shall be inserted directly after the bigword. If some other file is completely matched, a single `<space>` character shall be inserted after the bigword. After this operation, `sh` shall be placed in insert mode.

`*` Perform pathname expansion on the current bigword and insert all expansions into the command to replace the current bigword, with each expansion separated by a single `<space>` character. If at the end of the line, the current cursor position shall be moved to the first column position following the expansions and `sh` shall be placed in insert mode. Otherwise, the current cursor position shall be the last column position of the first character after the expansions and `sh` shall be placed in insert mode. If the current bigword contains none of the characters `?`, `*`, or `[`, before the operation, an asterisk shall be implicitly assumed at

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

1325 the end.

1326 *@letter* Insert the value of the alias named *_letter*. The symbol *letter* 8
1327 represents a single alphabetic character from the portable char- 8
1328 acter set; implementations may support additional characters as 8
1329 an extension. If the alias *letter* contains other editing com- 8
1330 mands, these commands shall be performed as part of the inser-
1331 tion. If no alias *letter* is enabled, this command shall have no
1332 effect.

1333 [*count*]~ Convert, if the current character is a lowercase letter, to the
1334 equivalent uppercase letter, and vice versa, as prescribed by the
1335 current locale. The current cursor position then shall be
1336 advanced by one character. If the cursor was positioned on the
1337 last character of the line, the case conversion shall occur, but the
1338 cursor shall not advance. If the ~ command is preceded by a
1339 *count*, that number of characters shall be converted, and the cur- 7
1340 sor shall be advanced to the character position after the last 7
1341 character converted. If the *count* is larger than the number of 7
1342 characters after the cursor, this shall not be considered an error;
1343 the cursor shall advance to the last character on the line.

1344 [*count*]. Repeat the most recent nonmotion command, even if it was exe- 7
1345 cuted on an earlier command line. If the previous command was 7
1346 preceded by a *count*, and no count is given on the . command,
1347 the count from the previous command shall be included as part
1348 of the repeated command. If the . command is preceded by a
1349 *count*, this shall override any *count* argument to the previous
1350 command. The *count* specified in the . command shall become 7
1351 the count for subsequent . commands issued without a count. 7

1352 [*number*]v Invoke the vi editor to edit the current command line in a tem-
1353 porary file. When the editor exits, the commands in the tem-
1354 porary file shall be executed. If a *number* is prefixed to the com-
1355 mand, it specifies the command number in the command history 8
1356 to be edited, rather than the current command line.

1357 [*count*]l (ell)
1358 [*count*]<space>
1359 Move the current cursor position to the next character position.
1360 If the cursor was positioned on the last character of the line, the
1361 terminal shall be alerted and the cursor shall not be advanced.
1362 If the *count* is larger than the number of characters after the
1363 cursor, this shall not be considered an error; the cursor shall
1364 advance to the last character on the line.

1365 [*count*]h Move the current cursor position to the *count*-th (default 1) pre-
1366 vious character position. If the cursor was positioned on the first
1367 character of the line, the terminal shall be alerted and the cursor
1368 shall not be moved. If the count is larger than the number of
1369 characters before the cursor, this shall not be considered an

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

error; the cursor shall move to the first character on the line.

[*count*]w — Move to the start of the next word. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the *count* is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[*count*]W — Move to the start of the next bigword. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the *count* is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[*count*]e — Move to the end of the current word. If at the end of a word, move to the end of the next word. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the *count* is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[*count*]E — Move to the end of the current bigword. If at the end of a bigword, move to the end of the next bigword. If the cursor was positioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the *count* is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[*count*]b — Move to the beginning of the current word. If at the beginning of a word, move to the beginning of the previous word. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the *count* is larger than the number of words preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line.

[*count*]B — Move to the beginning of the current bigword. If at the beginning of a bigword, move to the beginning of the previous bigword. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the *count* is larger than the number of bigwords preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line.

^ — Move the current cursor position to the first character on the input line that is not a <blank>.

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

1413 `$` Move to the last character position on the current command line.

1414 `0` (Zero.) Move to the first character position on the current com-
1415 mand line.

1416 [*count*]| Move to the *count* -th character position on the current command
1417 line. If no number is specified, move to the first position. The
1418 first character position shall be numbered 1. If the count is
1419 larger than the number of characters on the line, this shall not
1420 be considered an error; the cursor shall be placed on the last
1421 character on the line.

1422 [*count*]f*c* Move to the first occurrence of the character *c* that occurs after
1423 the current cursor position. If the cursor was positioned on the
1424 last character of the line, the terminal shall be alerted and the
1425 cursor shall not be advanced. If the character *c* does not occur in
1426 the line after the current cursor position, the terminal shall be
1427 alerted and the cursor shall not be moved.

1428 [*count*]F*c* Move to the first occurrence of the character *c* that occurs before
1429 the current cursor position. If the cursor was positioned on the
1430 first character of the line, the terminal shall be alerted and the
1431 cursor shall not be moved. If the character *c* does not occur in
1432 the line before the current cursor position, the terminal shall be
1433 alerted and the cursor shall not be moved.

1434 [*count*]t*c* Move to the character before the first occurrence of the character
1435 *c* that occurs after the current cursor position. If the cursor was
1436 positioned on the last character of the line, the terminal shall be
1437 alerted and the cursor shall not be advanced. If the character *c*
1438 does not occur in the line after the current cursor position, the
1439 terminal shall be alerted and the cursor shall not be moved.

1440 [*count*]T*c* Move to the character after the first occurrence of the character *c*
1441 that occurs before the current cursor position. If the cursor was
1442 positioned on the first character of the line, the terminal shall be
1443 alerted and the cursor shall not be moved. If the character *c*
1444 does not occur in the line before the current cursor position, the
1445 terminal shall be alerted and the cursor shall not be moved.

1446 [*count*]; Repeat the most recent `f`, `F`, `t`, or `T` command. Any number
1447 argument on that previous command shall be ignored. Errors
1448 are those described for the repeated command.

1449 [*count*], Repeat the most recent `f`, `F`, `t`, or `T` command, ignoring the
1450 number argument to that command. However, reverse the direc-
1451 tion of that command.

1452 `a` Enter insert mode after the current cursor position. Characters
1453 that are entered shall be inserted before the next character.

1454 `A` Enter insert mode after the end of the current command line.

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

1455 `i` Enter insert mode at the current cursor position. Characters
1456 that are entered shall be inserted before the current character.

1457 `I` Enter insert mode at the beginning of the current command line.

1458 `R` Enter insert mode, replacing characters from the command line
1459 beginning at the current cursor position.

1460 [*count*]`c` *motion*
1461 Delete the characters between the current cursor position and
1462 the cursor position that would result from the specified *motion*
1463 command. Then enter insert mode before the first character fol-
1464 lowing any deleted characters. If *count* is specified, it shall be
1465 applied to the *motion* command. A *count* shall be ignored for the
1466 motion commands:

1467 `0 ^ $ c`

1468 If the *motion* command is the character `c`, the current command
1469 line shall be cleared and insert mode shall be entered. If the
1470 *motion* command would move the current cursor position toward
1471 the beginning of the command line, the character under the
1472 current cursor position shall not be deleted. If the motion com-
1473 mand would move the current cursor position toward the end of
1474 the command line, the character under the current cursor posi-
1475 tion shall be deleted. If the *count* is larger than the number of
1476 characters between the current cursor position and the end of
1477 the command line toward which the motion command would
1478 move the cursor, this shall not be considered an error; all of the
1479 remaining characters in the aforementioned range shall be
1480 deleted and insert mode shall be entered. If the motion com-
1481 mand is invalid, the terminal shall be alerted, the cursor shall
1482 not be moved, and no text shall be deleted.

1483 `C` Delete from the current character to the end of the line and
1484 enter insert mode at the new end of line.

1485 `S` Clear the entire current command line and enter insert mode.

1486 [*count*]`r`*c* Replace the current character with the character *c*. With a
1487 number *count* , replace the current and the following *count* −1
1488 characters. After this command, the current cursor position
1489 shall be on the last character that was changed. If the *count* is
1490 larger than the number of characters after the cursor, this shall
1491 not be considered an error; all of the remaining characters shall
1492 be changed.

1493 [*count*]`_` Append a `<space>` after the current character position and then
1494 append the last bigword in the previous input line after the
1495 `<space>`. Then enter insert mode after the last character just
1496 appended. With a number *count* , append the *count* -th bigword
1497 in the previous line.

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

[*count*]x Delete the character at the current cursor position and place the deleted character(s) in the save buffer. If the cursor was positioned on the last character of the line, the character shall be deleted and the cursor position shall be moved to the previous character (the new last character). If the *count* is larger than the number of characters after the cursor, this shall not be considered an error; all the characters from the cursor to the end of the line shall be deleted.

[*count*]X Delete the character before the current cursor position and place the deleted character(s) in the save buffer. The character under the current cursor position shall not change. If the cursor was positioned on the first character of the line, the terminal shall be alerted, and the X command shall have no effect. If the line contained a single character, the X command shall have no effect. If the line contained no characters, the terminal shall be alerted and the cursor shall not be moved. If the *count* is larger than the number of characters before the cursor, this shall not be considered an error; all the characters from before the cursor to the beginning of the line shall be deleted.

[*count*]d *motion*
Delete the characters between the current cursor position and the character position that would result from the *motion* command. A number *count* repeats the *motion* command *count* times. If the *motion* command would move toward the beginning of the command line, the character under the current cursor position shall not be deleted. If the *motion* command is d, the entire current command line shall be cleared. If the *count* is larger than the number of characters between the current cursor position and the end of the command line toward which the motion command would move the cursor, this shall not be considered an error; all of the remaining characters in the aforementioned range shall be deleted. The deleted characters shall be placed in the save buffer.

D Delete all characters from the current cursor position to the end of the line. The deleted characters shall be placed in the save buffer.

[*count*]y *motion*
Yank (i.e., copy) the characters from the current cursor position to the position resulting from the *motion* command into a save buffer. A number *count* shall be applied to the *motion* command. If the *motion* command would move toward the beginning of the command line, the character under the current cursor position shall not be included in the set of yanked characters. If the *motion* command is y, the entire current command line shall be yanked into the save buffer. The current cursor position shall be unchanged. If the *count* is larger than the number of characters

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

between the current cursor position and the end of the command line toward which the motion command would move the cursor, this shall not be considered an error; all of the remaining characters in the aforementioned range shall be yanked.

`Y` Yank the characters from the current cursor position to the end of the line into the save buffer. The current character position shall be unchanged.

[*count*]`p` Put a copy of the current contents of the save buffer after the current cursor position. The current cursor position shall be advanced to the last character put from the save buffer. A *count* shall indicate how many copies of the save buffer shall be put.

[*count*]`P` Put a copy of the current contents of the save buffer before the current cursor position. The current cursor position shall be moved to the last character put from the save buffer. A *count* shall indicate how many copies of the save buffer shall be put.

`u` Undo the last command that modified the text of the current command line.

`U` Undo all changes made to the current command line since first entering command mode on the line.

[*count*]`k`
[*count*]– Replace the current command line with the previous command line in the shell command history. The cursor shall be positioned on the first character of the new command. A count preceding the command shall have the same effect as executing the command *count* times. If a `k` or `-` command retreats past the maximum number of commands in effect for this shell (affected by the **HISTSIZE** environment variable), the terminal shall be alerted and the command shall have no effect.

[*count*]`j`
[*count*]`+` Replace the current command line with the next command line in the shell command history. The cursor shall be positioned on the first character of the new command. The command history 8
position shall be remembered, and any `k` or – command, or `j` or `+` command, shall decrement or increment that position and then shall fetch the line at the new position. If a `j` or `+` command advances past the most recent line in the history, the current command line shall be restored to the contents before the first `k` or –. 7

[*number*]`G` Replace the current command line with the contents of the oldest command line stored in the shell command history. With a number *number* , replace the current command line with the contents of command *number* in the history.

/string `<newline>`
Move backward through the command history, searching for the

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

specified *string* , beginning with the previous command line. If it is not found, the current command line shall be unchanged. If it is found in a previous line, this command shall behave equivalently to a set of `k` commands to reach that line. If *string* begins with ^, the characters after the ^ shall be matched only at the beginning of a line.

`?`*string* `<newline>`
: Move forward through the command history, searching for the specified string. If it is not found, the current command line shall be unchanged. If the string is found in the current command line, the current cursor position shall be moved to the beginning of that string. If it is found in the history, this command shall behave equivalently to a set of `j` commands to reach that line. If *string* begins with ^, the characters after the ^ shall be matched only at the beginning of a line.

`n`
: Repeat the most recent / or ? command.

`N`
: Repeat the most recent / or ? command, reversing the direction of the search.

4.56.10 `sh` Rationale. *(This subclause is not a part of P1003.2a)*

The default messages for the various **MAIL**-related messages are unspecified because they vary across implementations. Typical messages are:

```
you have mail\n    or    you have new mail\n
```

It is important that the descriptions of command line editing refer to the same shell as that in the base standard so that interactive users can also be application programmers without having to deal with programmatic differences in their two environments. It is also essential that the utility name `sh` be specified because this explicit utility name is too firmly rooted in historical practice of application programs for it to change.

Consideration was given to mandating a diagnostic message when attempting to set vi-mode on terminals that do not support command line editing. However, it is not historical practice for the shell to be cognizant of all terminal types and thus be able to detect inappropriate terminals in all cases. Implementations are encouraged to supply diagnostics in this case whenever possible, rather than leaving the user in a state where editing commands work incorrectly.

In earlier drafts, the KornShell-derived `emacs` mode of command-line editing was included, even though the `emacs` editor itself was not. The community of `emacs` editing enthusiasts was quite adamant that the full `emacs` editor not be included in POSIX.2, because it was concerned that an attempt to standardize this very powerful environment would encourage vendors to ship versions conforming strictly to the standard, but lacking the extensibility required by the community.

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

The author of the original `emacs` program also expressed his desire to omit the program. Furthermore, there were a number of historical systems that did not include `emacs`, or included it without supporting it, but there were very few that did not include and support `vi`. The shell `emacs` command-line editing mode was finally omitted from this standard because it became apparent that the KornShell version and the editor being distributed with the GNU system had diverged in some respects. The author of `emacs` requested that the POSIX `emacs` mode either be deleted or have a significant number of unspecified conditions. Although the KornShell author agreed to consider changes to bring the shell into alignment, the developers of the standard decided to defer specification at this time, rather than attempting to agree on a specific subset of `emacs` late within balloting of the standard. (Furthermore, the standard developers included few individuals who had direct experience with `emacs` mode and the specification did not achieve the level of review afforded to the `vi` mode.) It is assumed that the `emacs` and KornShell developers will converge on a definition acceptable to both groups and this may be used as a model for a future version of this standard. In the interim, implementations are free to offer additional command-line editing modes based on the exact models of editors their users are most comfortable with.

4.70 `wait` — Await Process Completion

⇒ **4.70 `wait` Operands.** *In subclause 4.70, modify the description of the pid operand in the Operands subclause:*

pid One of the following:

(1) *<current contents>*

(2) On systems supporting both the POSIX .1 {8} job control option and the User Portability Utilities Option, a job control job ID (see 2.2.2.206) that identifies a background process group to be waited for. The job control job ID notation is applicable only for invocations of `wait` in the current shell execution environment; see 3.12. The exit status of `wait` shall be determined by the last command in the pipeline.

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

This additional syntax for *pid* is applicable only to systems having the job control features of the User Portability Extension. Thus, it is described in this supplement document, rather than in the base POSIX. 2 document. See the rationale for `kill` for additional information on the effects of the job control job ID notation outside the current shell execution environment.

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

⇒ **4.70.10 `wait` Rationale.** *In subclause 4.70, add the following example to the rationale.*

Example: If the following sequence of commands is run in less than 31 seconds:

```
sleep 257 | sleep 31 &
jobs -l %%
```

either of the following commands will return the exit status of the second `sleep` in the pipeline:

```
wait <pid of sleep 31>
wait %%
```

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

Section 5: User Portability Utilities Option

This section describes the utilities that shall be implemented on all systems that claim conformance to the User Portability Utilities Option. Certain utilities are noted as having features that cannot be implemented on all terminal types; if the {POSIX 2_CHAR _TERM } option is supported, the system shall support all such features on at least one terminal type. See 2.14.

Some of the utilities are only required on systems that also support the Software Development Utilities Option, the character-at-a-time terminal option (see 2.14) or that support the POSIX .1 {8} job control option; such utilities are so marked, directly before their Synopsis subclauses. All of the other utilities in this section are required only on systems that claim conformance to the User Portability Utilities Option.

User Portability Utilities Option Rationale. *(This subclause is not a part of P1003.2a)* .

All of the new utilities defined in the UPE are collected into a single large section. This was done for the practical reason that empty sections (chapters) were being held open for this work in the base POSIX. 2 document. It was considered impractical to have enough empty chapters to handle all of the permutations of UPE , software development, and special terminal characteristics.

A summary of the combinations of options required for certain utilities in this section (where • indicates that the entire utility is optional unless the feature is provided and text indicates the portions that are optional):

Utility	{_POSIX 2 _ _UPE }	{_POSIX 2 SW _DEV }	{_POSIX _JOB _CONTROL }	{_POSIX 2 _CHAR _TERM }
bg	•		•	
ex	•	-t option		see NOTE
ctags	•	•		
fg	•		•	
more	•	-t option		see NOTE
nm	•	•		
sh	•			see NOTE
talk	•			see NOTE
vi	•	-t option		•

NOTE: The ex, more, sh, and talk utilities also have terminal-related requirements that are allowed to result in error messages on some terminals (or all terminals, if {POSIX 2_CHAR _TERM } is

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

not supported); see their descriptions.

5.1 `alias` — Define or display aliases

5.1.1 Synopsis

`alias [` *alias-name* `[=`*string* `] ... ]`

5.1.2 Description

The `alias` utility shall create or redefine alias definitions or write the values of existing alias definitions to standard output. An alias definition provides a string value that shall replace a command name when it is encountered. See 3.3.1.

An alias definition shall affect the current shell execution environment and the execution environments of the subshells of the current shell. When used as specified by this standard, the alias definition shall not affect the parent process of the current shell nor any utility environment invoked by the shell. See 3.12.

5.1.3 Options

None.

5.1.4 Operands

The following operands shall be supported by the implementation:

alias-name Write the alias definition to standard output.

alias-name `=`*string*
Assign the value of *string* to the alias *alias-name* .

If no operands are given, all alias definitions shall be written to standard output.

5.1.5 External Influences

5.1.5.1 Standard Input

None.

5.1.5.2 Input Files

None.

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

5.1.5.3 Environment Variables

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

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.

5.1.5.4 Asynchronous Events

Default.

5.1.6 External Effects

5.1.6.1 Standard Output

The format for displaying aliases (when no operands or only *name* operands are specified) shall be:

"%s=%s\n", *name* , *value*

The *value* string shall be written with appropriate quoting so that it is suitable for re-input to the shell. See the description of shell quoting in 3.2.

5.1.6.2 Standard Error

Used only for diagnostic messages.

5.1.6.3 Output Files

None.

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

5.1.7 Extended Description

None.

5.1.8 Exit Status

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

0 Successful completion.

>0 One of the *name* operands specified did not have an alias definition, or an error occurred.

5.1.9 Consequences of Errors

Default.

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

Usage, Examples

None.

History of Decisions Made

The `alias` description is based on that from historical KornShell implementations. Known differences exist between that and the C-Shell. The KornShell version was adopted to be consistent with all the other KornShell features in POSIX. 2, such as command-line editing.

Since `alias` affects the current shell execution environment, it is generally provided as a shell regular built-in.

Historical versions of the KornShell have allowed aliases to be exported to scripts that are invoked by the same shell. This is triggered by the `alias –x` flag; it is allowed by POSIX. 2 only when an explicit extension such as `–x` is used. The working group felt that aliases were of use primarily to interactive users and that they should normally not affect shell scripts called by those users; functions are available to such scripts.

Historical versions of the KornShell have not written aliases in a quoted manner suitable for reentry to the shell, but the base POSIX. 2 standard has made this a requirement for all similar output. Therefore, consistency with the base standard was chosen over this minor detail of historical practice.

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

5.2 `at` — Execute commands at a later time

5.2.1 Synopsis

`at [-m][-f` *file* `][-q` *queuename* `] -t` *time*

`at [-m][-f` *file* `][-q` *queuename* `]` *timespec* ...

`at -r` *at_job_id* ...

`at -l -q` *queuename*

`at -l[`*at_job_id* ...`]`

5.2.2 Description

The `at` utility shall read commands from standard input and group them together as an *at-job* , to be executed at a later time.

8

The at-job shall be executed in a separate invocation of the shell, running in a separate process group with no controlling terminal, except that the environment variables, current working directory, file creation mask, and other implementation-defined execution-time attributes in effect when the `at` utility is executed shall be retained and used when the at-job is executed.

When the at-job is submitted, the *at_job_id* and scheduled time shall be written to standard error. The *at_job_id* is an identifier that shall be a string consisting solely of alphanumeric characters and the period character. The *at_job_id* shall be assigned by the system when the job is scheduled such that it uniquely identifies a particular job.

User notification and the processing of the job's standard output and standard error are described under the `-m` option. 8 8

5.2.3 Options

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

The following options shall be supported by the implementation:

`-f` *file* Specify the pathname of a file to be used as the source of the at-job, instead of standard input. 8 8

`-l` (The letter ell.) Report all jobs scheduled for the invoking user if no *at_job_id* operands are specified. If *at_job_id* s are specified, report only information for these jobs. The output shall be written to standard output.

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

`-m` Send mail to the invoking user after the at-job has run, announcing its completion. Standard output and standard error produced by the at-job shall be mailed to the user as well, unless redirected elsewhere. Mail shall be sent even if the job produces no output.

If `-m` is not used, the job's standard output and standard error shall be provided to the user via an implementation-defined mechanism, unless they are redirected elsewhere; if there is no such output to provide, the implementation need not notify the user of the job's completion.

`-q` *queuename*
Specify in which queue to schedule a job for submission. When used with the `-l` option, limit the search to that particular queue. By default, at-jobs shall be scheduled in queue '`a`'. In contrast, queue '`b`' shall be reserved for batch jobs. (See the `batch` utility in 5.3.) The meanings of all other *queuenames* are implementation defined.

`-r` Remove the jobs with the specified *at_job_id* operands that were previously scheduled by the `at` utility.

`-t` *time* Submit the job to be run at the time specified by the *time* option-argument, which shall have the format as specified by the `touch` (see 4.63) utility.

5.2.4 Operands

The following operands shall be supported by the implementation:

at_job_id
The name reported by a previous invocation of the `at` utility at the time the job was scheduled.

timespec
Submit the job to be run at the date and time specified. All of the *timespec* operands shall be concatenated into the equivalent of a single string with `<space>` separation added between the operands and this string shall be parsed as described in the grammar at the end of this subclause. The date and time shall be interpreted as being in the time zone of the user (as determined by the **TZ** variable), unless a time-zone name appears as part of *time* , below.

In the POSIX Locale, the following describes the three parts of the time specification string. All of the values from the LC_TIME categories in the POSIX Locale shall be recognized in a case-insensitive manner.

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

time The *time* can be specified as one, two, or four digits. One- and two-digit numbers shall be taken to be hours, four-digit numbers to be hours and minutes. The time can alternately be specified as two numbers separated by a colon, meaning *hour* `:`*minute* . An AM/PM indication (one of the values from the `am_pm` keywords in the LC_TIME locale category) can follow the time; otherwise, a 24-hour clock time shall be understood. A time-zone name can also follow to further qualify the time. The acceptable time-zone names are implementation defined, except that they shall be case-insensitive and the string `"utc"` shall be supported to indicate the time is in Coordinated Universal Time. The *time* field can also be one of the following tokens in the POSIX Locale:

`midnight`
: Indicates the time 12:00 am (00:00).

`noon` Indicates the time 12:00 pm.

`now` Indicate the current day and time. Invoking `at` <*now*> shall submit an at-job for potentially immediate execution (i.e., subject only to unspecified scheduling delays).

date An optional *date* can be specified as either a month name (one of the values from the `mon` or `abmon` keywords in the LC_TIME locale category) followed by a day number (and possibly year number preceded by a comma) or a day of the week (one of the values from the `day` or `abday` keywords in the LC_TIME locale category). Two special ‘‘days’’ shall be recognized in the POSIX Locale:

`today`
: Indicates the current day.

`tomorrow`
: Indicates the day following the current day.

If no *date* is given, `today` shall be assumed if the given time is greater than the current time and `tomorrow` shall be assumed if it is less. If the given month is less than the current month (and no year is given), next year shall be assumed.

increment

The optional *increment* shall be a number preceded by a plus sign (+) and suffixed by one of the following: `minutes`, `hours`, `days`, `weeks`, `months`, or `years`. (The singular forms shall be also accepted.) The keyword `next` shall be equivalent to an increment number of `+ 1`. For example, the following are equivalent commands:

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

```
                    at 2pm + 1 week
                    at 2pm next week
```

The following grammar describes the precise format of *timespec* in the POSIX Locale. The general conventions for this style of grammar are described in 2.1.2. Any discrepancies found between this grammar and the preceding description shall be resolved in favor of the grammar. The longest possible token or delimiter shall be recognized at a given point. White space shall also delimit tokens.

```
%token hr24clock_hr_min
%token hr24clock_hour
/*
  A hr24clock_hr_min is a one, two, or four digit number.
  A one or two digit number constitutes a hr24clock_hour.
  A hr24clock_hour may be any of the single digits '0' - '9',
  or may be double digits, ranging from "00" - "23".
  If a hr24clock_hr_min is a four digit number, the first
  two digits must be a valid hr24clock_hour, while the last
  two represent the number of minutes, from "00" - "59".
*/
%token wallclock_hr_min
%token wallclock_hour
/*
  A wallclock_hr_min is a one, two or four digit number.
  A one or two digit number constitutes a wallclock_hour.
  A wallclock_hour may be any of the single digits '1' - '9',
  or may be double digits, ranging from "01" - "12".
  If a wallclock_hr_min is a four digit number, the first two
  digits must be a valid wallclock_hour, while the last two
  represent the number of minutes, from "00" - "59".
*/
%token minute
/*
  A minute is a one or two digit number whose values can be
  '0' - '9' or "00" - "59".
*/
%token day_number
/*
  A day_number is a number in the range appropriate for the
  particular month and year specified by month_name and
  year_number, respectively.  If no year_number is given,
  the current year is assumed if the given date and time
  are later this year.  If no year_number is given and the
  date and time have already occurred this year and the
  month is not the current month, next year is the assumed year.
*/
%token year_number
/*
  A year_number is a four-digit number representing the
  year A.D., in which the at job is to be run.
```

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