IEEE P1003.2a Draft 8 – December 1991

The C-shell has a built-in version of `nice` that has a different interface from the one described here.

The term ''utility'' is used, rather than ''command,'' to highlight the fact that shell compound commands, pipelines, etc., cannot be used. Special built-ins also cannot be used. However, ''utility'' includes user application programs and shell scripts, not just utilities defined in this standard.

Historical implementations of `nice` provide a priority range of 40 or 41 discrete steps, with the default priority being the midpoint of that range. By default they lower the priority of the executed utility by 10.

Some historical documentation states that the *increment* value must be within a fixed range. This is misleading; the valid *increment* values on any invocation are determined by the current process priority, which is not always the default.

The two limits in a previous draft, {POSIX2_MAX_NICE} and {POSIX2_MIN_NICE}, were removed when the `nice` description was reworded to allow users to exceed the system's range with impunity.

The definition of *system scheduling priority* is not intended to suggest that all 7
processes in a system have priorities that are comparable. Scheduling policy 7
extensions such as the realtime priorities in POSIX.4 make the notion of a single 7
underlying priority for all scheduling policies problematic. Some systems may 7
implement the `nice`-related features to affect all processes on the system, others 7
to affect just the general time-sharing activities implied by POSIX.2, and others 7
may have no effect at all. Because of the use of ''implementation defined'' in `nice` 7
and `renice`, a wide range of implementation strategies are possible. 7

5.21 nm — Write the name list of an object file

This utility shall be provided on systems that support both the User Portability 7
Utilities Option and the Software Development Utilities Option. On other sys- 7
tems it is optional. 7

5.21.1 Synopsis

`nm [−APv] [ −g | −u ] [−t` *format* `]` *file* . . .

5.21.2 Description

The `nm` utility shall display symbolic information appearing in the object file, executable file, or object-file library named by *file*. If no symbolic information is available for a valid input file, the `nm` utility shall report that fact, but not consider that an error condition.

The default base used when numeric values are written is unspecified.

5.21.3 Options

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

The following options shall be supported by the implementation:

`−A` Write the full pathname or library name of an object on each line.

`−g` Write only external (global) symbol information.

`−P` Write information in a portable output format, as specified in 5.21.6.1.

`−t` *format* Write each numeric value in the specified format. The format shall be dependent on the single character used as the *format* option-argument:

- `d` The offset shall be written in decimal.
- `o` The offset shall be written in octal.
- `x` The offset shall be written in hexadecimal.

`−u` Write only undefined symbols.

`−v` Sort output by value instead of alphabetically.

5.21.4 Operands

The following operand shall be supported by the implementation:

file A pathname of an object file, executable file, or object-file library.

5.21.5 External Influences

5.21.5.1 Standard Input

See Input Files.

5.21.5.2 Input Files

The input file shall be an object file, an object-file library whose format is the 7
same as those produced by the `ar` utility for link editing (see 6.1), or an execut- 7
able file. The *nm* utility may accept additional implementation-defined object 7
library formats for the input file. 7

5.21.5.3 Environment Variables

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

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 for the symbol-name and symbol-value collation sequences.

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). 7

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

5.21.5.4 Asynchronous Events

Default.

5.21.6 External Effects

5.21.6.1 Standard Output

If symbolic information is present in the input file(s), then for each file, or for each 7
member of an archive, the `nm` utility shall write the following information to stan- 7
dard output. By default, the format is unspecified, but the output shall be sorted alphabetically by symbol name.

— Library or object name, if `−A` is specified.

— Symbol name.

— Symbol type, which shall either be one of the following single characters or 7
an implementation-defined type represented by a single character: 7

`A`	Global absolute symbol
`a`	Local absolute symbol
`B`	Global ''bss''(i.e., uninitialized data space) symbol
`b`	Local bss symbol
`D`	Global data symbol
`d`	Local data symbol
`T`	Global text symbol
`t`	Local text symbol
`U`	Undefined symbol

— Value of the symbol.

— The size associated with the symbol, if applicable.

This information may be supplemented by additional information specific to the implementation.

If the `−P` option is specified, the previous information shall be displayed using the following portable format. The three versions differ depending on whether `-t d`, `-t o`, or `-t x` was specified, respectively:

`"%s: %s %s %d %d\n"`, *<library/object name>* , *<name>* , *<type>*, *<value>* , *<size>*

`"%s: %s %s %o %o\n"`, *<library/object name>* , *<name>* , *<type>*, *<value>* , *<size>*

`"%s: %s %s %x %x\n"`, *<library/object name>* , *<name>* , *<type>*, *<value>* , *<size>*

where *<library/object name>* and its trailing colon and `<blank>`s are omitted when `−A` is not specified and *<size>* and its preceding `<blank>`s are omitted when inapplicable to the symbol type.

5.21.6.2 Standard Error

Used only for diagnostic messages.

5.21.6.3 Output Files

None.

5.21.7 Extended Description

None.

5.21.8 Exit Status

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

- 0 Successful completion.
- >0 An error occurred.

5.21.9 Consequences of Errors

Default.

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

Usage, Examples

Historical implementations of `nm` have used different bases for numeric output and supplied different default types of symbols that were reported. The `−t` *format* option, similar to that used in `od` and `strings`, can be used to specify the numeric base; `−g` and `−u` can be used to restrict the amount of output or the types of symbols included in the output.

History of Decisions Made

The option list was significantly reduced from that provided by historical implementations.

The `nm` description is a subset of both the System V and BSD `nm` utilities with no specified default output. 7

It was recognized that mechanisms for dynamic linking make this utility less meaningful when applied to a library, but the value of `nm` during software development was judged to outweigh other limitations.

The compromise of using `−t` *format* versus using `−d`, `−o`, and other similar options was necessary because of differences in the meaning of `−o` between implementations. The `−o` option from BSD has been provided here as `−A` to avoid collision with the `−o` from System V (which has been provided here as `-t o`).

The default output format of `nm` is not specified because of differences in historical implementations. The `P` option was added to satisfy balloting objections relating to the use of `nm` in scripts as well as interactively. Some type of portable format was required. The format given in 5.21.6.1 uses spaces between the fields, which may be any number of `<blank>`s required to get the columns to line up. The single-character types were selected to match historical practice and the requirement that implementation additions also be single characters make parsing the information easier for shell scripts.

5.22 `patch` — Apply changes to files

5.22.1 Synopsis

`patch` [`−blNR`] [`−c` | `−e` | `−n`] [`−d` *dir*] [`−D` *define*] [`−i` *patchfile*] [`−o` *outfile*] 8
[`−r` *rejectfile*] [*file*] 8

5.22.2 Description

The `patch` utility shall read a source (‘‘patch’’) file containing any of the three forms of difference (‘‘diff’’) listings produced by the `diff` (see 4.17) utility (normal, context, or in the style of `ed`; see 4.20) and apply those differences to a file. By default, `patch` shall read from the standard input.

The `patch` utility shall attempt to determine the type of the diff listing, unless overruled by a `−c`, `−e`, or `−n` option.

If the patch file contains more than one patch, `patch` shall attempt to apply each
of them as if they came from separate patch files. (In this case the name of the 7
patch file must be determinable for each `diff` listing.) 7

5.22.3 Options

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

The following options shall be supported by the implementation:

`−b` Save a copy of the original contents of each modified file, before 7
the differences are applied, in a file of the same name with the 7
suffix `.orig` appended to it. If the file already exists, it shall be 7
overwritten; if multiple patches are applied to the same file, the 7
`.orig` file shall be written only for the first patch. When the 8
`−o` *outfile* option is also specified, *file* `.orig` shall not be created 7
but, if *outfile* already exists, *outfile* `.orig` shall be created. 7

`−c` Interpret the patch file as a context diff (the output of the utility `diff` when the `−c` or `−C` options are specified).

`−d` *dir* Change the current directory to *dir* before processing as described in 5.22.7.

`−D` *define* Mark changes with the C preprocessor construct:

```
#ifdef define
...
#endif
```

The option-argument *define* shall be used as the differentiating symbol.

`−e` Interpret the patch file as an `ed` script, rather than a `diff` script.

`−i` *patchfile* Read the patch information from the file named by the pathname *patchfile* , rather than the standard input.

`−l` (The letter ell.) Cause any sequence of `<blank>`s in the diff script to match any sequence of `<blank>`s in the input file. Other characters shall be matched exactly.

`−n` Interpret the script as a normal diff.

`−N` Ignore patches where the differences have already been applied to 7
the file; by default, already-applied patches shall be rejected. 7

`−o` *outfile* Instead of modifying the files (specified by the *file* operand or the 7
diff listings) directly, write a copy of the file referenced by each 7
patch, with the appropriate differences applied, to *outfile* . Multi- 7
ple patches for a single file shall be applied to the intermediate 7
versions of the file created by any previous patches, and shall 7
result in multiple, concatenated versions of the file being written 7
to *outfile* . 7

8

`−R` Reverse the sense of the patch script; i.e., assume that the diff script was created from the new version to the old version. The `−R` option cannot be used with `ed` scripts. The `patch` utility attempts to reverse each portion of the script before applying it. Rejected differences shall be saved in swapped format. If this option is not specified, and until a portion of the patch file is successfully applied, `patch` attempts to apply each portion in its ''reversed'' sense as well as in its normal sense. If the attempt is successful, the user shall be prompted to determine if the `−R` option should be set.

`−r` *rejectfile* Override the default reject file name. In the default case, the 7
reject file shall have the same name as the output file, with the 7
suffix `.rej` appended to it. See 5.22.7.3. 7

5.22.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to patch.

5.22.5 External Influences

5.22.5.1 Standard Input

See Input Files.

5.22.5.2 Input Files

Input files shall be text files.

5.22.5.3 Environment Variables

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

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

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

LC_MESSAGES This variable shall determine the processing of affirmative responses and the language in which messages should be written.

LC_TIME This variable shall determine the locale for recognizing the format of file time stamps written by the `diff` utility in a context-diff input file.

5.22.5.4 Asynchronous Events

Default.

5.22.6 External Effects

5.22.6.1 Standard Output

None.

5.22.6.2 Standard Error

Used for diagnostic and informational messages. 7

5.22.6.3 Output Files

The output of the `patch` utility, the save files (`.orig` suffixes) and the reject files (`.rej` suffixes) shall be text files.

5.22.7 Extended Description

A patchfile may contain patching instructions for more than one file; file names 8
shall be determined as specified in 5.22.7.2. When the `-b` option is specified, for each patched file, the original shall be saved in a file of the same name with the suffix `.orig` appended to it.

For each patched file, a reject file may also be created as noted in 5.22.7.3. In the absence of a `-r` option, the name of this file shall be formed by appending the suffix `.rej` to the original file name.

5.22.7.1 Patchfile Format

The patch file shall contain zero or more lines of header information followed by one or more patches. Each patch shall contain zero or more lines of filename identification in the format produced by `diff -c` (see 4.17), and one or more sets of `diff` output, which are customarily called ''hunks.''

The `patch` utility shall recognize the following expression in the header information:

`Index:` *pathname*	The file to be patched is named *pathname* .

8

If all lines (including headers) within a patch begin with the same leading 7
sequence of `<blank>`s, the `patch` utility shall remove this sequence before 7
proceeding. Within each patch, if the type of diff is context, the `patch` utility shall recognize the following expressions:

`***` *filename timestamp*	The patches arose from *filename.*
`---` *filename timestamp*	The patches should be applied to *filename.*

Each hunk within a patch shall be the `diff` output to change a line range within the original file. The line numbers for successive hunks within a patch shall occur in ascending order.

5.22.7.2 Filename Determination

If no *file* operand is specified, `patch` shall perform the following steps to obtain a pathname:

(1) If the patch contains the strings *** and `---`, the `patch` utility shall test 8
for the existence of each pathname relative to the current directory. If 8
both files exist, the `patch` utility shall assume that no pathname can be 7
obtained from this step. 7

(2) If the header information contains a line with the string `Index:`, the
`patch` utility shall test for the existence of this file relative to the current 8
directory. 8

(3) If no pathname can be obtained by applying the steps (1) or (2), or if the
pathnames obtained do not exist, the `patch` utility shall write a prompt 8
to standard output and request a filename interactively from standard 8
input.

5.22.7.3 Patch Application

If the `-c`, `-e`, or `-n` option is present, the `patch` utility shall interpret information within each hunk as a context diff, an `ed` diff, or a normal diff, respectively. In the absence of any of these options, the `patch` utility shall determine the type of diff based on the format of information within the hunk.

8

For each hunk, the `patch` utility shall begin to search for the place to apply the
patch at the line number at the beginning of the hunk, plus or minus any offset
used in applying the previous hunk. If lines matching the hunk context are not 7
found, the `patch` utility shall scan both forwards and backwards at least 1000 7
bytes for a set of lines that match the hunk context. 7

If no such place is found, and it is a context diff, then another scan shall take place, ignoring the first and last line of context. If that fails, the first two and last two lines of context shall be ignored and another scan shall be made. Implementations may search more extensively for installation locations.

If no location can be found, the `patch` utility shall append the hunk to the reject 7
file. The rejected hunk shall be written in context-diff format regardless of the
format of the patch file. If the input was a normal or `ed`-style diff, the reject file 7
may contain diffs with zero lines of context. The line numbers on the hunks in
the reject file may be different than the line numbers in the patch file since they
shall reflect the approximate locations for the failed hunks in the new file rather
than the old one.

If the type of patch is an `ed` diff, the implementation may accomplish the patching by invoking the `ed` utility.

5.22.8 Exit Status

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

- 0 Successful completion.
- 1 One or more lines were written to a reject file.
- >1 An error occurred.

5.22.9 Consequences of Errors

Patches that cannot be correctly placed in the file shall be written to a reject file.

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

Usage, Examples

The −R option will not work with `ed` scripts because there is too little information to reconstruct the reverse operation.

8

History of Decisions Made

Some of the functionality in historical `patch` implementations has not been specified. The following documents features present in historical implementations that have not been specified.

The `patch` utility's ability to ‘‘check out’’ or ‘‘get’’ suitable RCS or SCCS files if the source file is unavailable was deleted because POSIX.2 has not yet, if it ever will, required SCCS or RCS, and, more importantly, because it violates the ‘‘Principle of Least Astonishment’’ for users. It is strongly suggested that this functionality, if provided, be under the control of a command-line option.

Another deleted piece of functionality was the ‘‘+’’ pseudo-option allowing an additional set of options and patch file operand to be given. This was seen as being insufficiently useful to standardize.

In historical implementations, if the string `Prereq:` appeared in the header, the 8
`patch` utility would search for corresponding version information (the string 8
specified in the header, delimited by <blank>s or the beginning or end of a line or 8
the file) anywhere in the original file. This was deleted as too simplistic and 8
insufficiently trustworthy a mechanism to standardize. For example, if 8

```
Prereq: 1.2
```
8

were in the header, the presence of a delimited 1.2 anywhere in the file would 8
satisfy the prerequisite. 8

The following options were dropped from historical implementations of `patch`, as insufficiently useful to standardize.

`−b` The `−b` option traditionally provided a method for changing the backup file's name extension from `.orig`. This option has been modified and retained in the current specification.

`−F` The `−F` option specified the number of lines of a context diff to ignore when searching for a place to install a patch.

`−f` The `−f` option traditionally caused `patch` to not request additional information from the user.

`−p` The `−p` option attempted to provide a method of customizing the patch file to local user file hierarchies.

Historical versions of `patch` resolved file names in a more complex fashion than that described by this standard. If no *file* operand was specified, the full pathnames and terminating filenames in any context diff as well as any pathnames specified using the `Index:` keyword in the patch header were checked for existence. If none of these files existed, the user was prompted for the name of a file to patch. Additionally, the `−p` option could be used to delete some of the pathname components from the context diff and `Index:` keyword pathnames. POSIX simplifies this behavior and does not require the `−p` option for two reasons. The first is that sites with strong naming and stylistic conventions, or with local and ''official'' copies of programs, could easily patch the wrong files with the user being unaware of the error. The second is that the `−p` option was only useful if the local tree was entirely consistent with the tree from which the patch file was created. This was believed unlikely, particularly in patch files with multiple patches, and the functionality is trivially replaced by creating the patch file at a more reasonable place in the tree.

`−r` The `−r` option traditionally provided a method of overriding the default reject file's extension from `.rej`.

`−s` The `−s` option traditionally caused `patch` to work silently unless an error occurred.

`−x` The `−x` option traditionally set internal debugging flags.

In some file system implementations, the saving of a `.orig` file may produce unwanted results. In the case of 12, 13, or 14-character filenames, on file systems supporting 14-character maximum filenames, the `.orig` file will overwrite the new file. The reject file may also exceed this filename limit. It was suggested, due to some existing practice, that a tilde (~) suffix be used instead of `.orig` and some other character instead of the `.rej` suffix. This was rejected because it is not obvious to the user which file is which. The suffixes `.orig` and `.rej` are clearer and more understandable.

The `−b` option has the opposite sense in some historical implementations—do not save the `.orig` file. The default case here is to not save the files, making `patch` behave more consistently with the other standard utilities.

The `−w` option in previous drafts was changed to `−l` to match historical practice.

The `−N` option was included because without it, a noninteractive application cannot reject previously applied patches. For example, if a user is piping the output of `diff` into the `patch` utility, and the user only wants to patch a file to a newer version noninteractively, the `−N` option is required.

Changes to the `−l` option description were proposed to allow matching across 8
`<newline>`s in addition to just `<blank>`s. Since this is not historical practice, 8
and since some ambiguities could result, it is suggested that future developments 8
in this area be triggered by another option letter, such as `−L`. 8

5.23 ps — Report process status

5.23.1 Synopsis

`ps` [−aA] [−G *grouplist*] [−o *format*] . . . [−p *proclist*] [−t *termlist*] 7
[−U *userlist*]

5.23.2 Description

The `ps` utility shall write information about processes, subject to having the appropriate privileges to obtain information about those processes.

When the `−o` option is not specified, information about processes selected shall be
written in an implementation-defined manner. By default, `ps` shall select all 7
processes with the same effective user ID as the current user and the same con- 7
trolling terminal as the invoker. In all cases, the selection of processes to be 7
reported shall be subject to the constraints of appropriate privileges. 7

5.23.3 Options

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

The following options shall be supported by the implementation.

`−a` Write information for all processes associated with terminals. 7
Implementations may omit session leaders from this list. 7

`−A` Write information for all processes.

`−G` *grouplist*
Write information for processes whose real group ID numbers are given in *grouplist* . The *grouplist* shall be a single argument in the form of a `<blank>`- or comma-separated list.

−o *format* Write information according to the format specification given in *format*. This is fully described in 5.23.6.1. Multiple **−o** options can be specified; the format specification shall be interpreted as the <space>-separated concatenation of all the *format* option-arguments.

−p *proclist* Write information for processes whose process ID numbers are given in *proclist*. The *proclist* shall be a single argument in the form of a <blank>- or comma-separated list.

−t *termlist* Write information for processes associated with terminals given in *termlist*. The *termlist* shall be a single argument in the form of a <blank>- or comma-separated list. Terminal identifiers shall be given in an implementation-defined format.

−U *userlist* Write information for processes whose real user ID numbers or login names are given in *userlist*. The *userlist* shall be a single argument in the form of a <blank>- or comma-separated list.

With the exception of −o *format*, all of the options shown are used to select 7
processes. If any are specified, the default list shall be ignored and ps shall select 7
the processes represented by the inclusive OR of all the selection-criteria options. 7

5.23.4 Operands

None.

5.23.5 External Influences

5.23.5.1 Standard Input

None.

5.23.5.2 Input Files

None.

5.23.5.3 Environment Variables

The following environment variables shall affect the execution of ps:

COLUMNS This variable shall override the system-selected horizontal 7
screen size, used to determine the number of text columns 7
to display. See 2.6 for valid values and results when it is 7
unset or null. 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_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.

LC_TIME This variable shall determine the format and contents of the date and time strings displayed.

5.23.5.4 Asynchronous Events

Default.

5.23.6 External Effects

5.23.6.1 Standard Output

When the `−o` option is not specified, the standard output format is implementation defined. The `−o` option allows the output format to be specified under user control.

The format specification shall be a list of names presented as a single argument, `<blank>`- or comma-separated. Each variable has a default header. The default header can be overridden by appending an equals sign and the new text of the header. The rest of the characters in the argument shall be used as the header text. The fields specified shall be written in the order specified on the command line, and should be arranged in columns in the output. The field widths shall be selected by the system to be at least as wide as the header text (default or overridden value). If the header text is null, such as `−o user=`, the field width shall be at least as wide as the default header text. If all header text fields are null, no header line shall be written.

The following names shall be recognized in the POSIX Locale: 8

`ruser` The real user ID of the process. This shall be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

`user` The effective user ID of the process. This shall be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

`rgroup` The real group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

`group` The effective group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

`pid` The decimal value of the process ID.

`ppid` The decimal value of the parent process ID.

`pgid` The decimal value of the process group ID.

`pcpu` The ratio of CPU time used recently to CPU time available in the 8
same period, expressed as a percentage. The meaning of 8
‘‘recently’’ in this context is unspecified. The CPU time available 8
is determined in an unspecified manner.

`vsz` The size of the process in (virtual) memory in kilobytes as a decimal integer.

`nice` The decimal value of the system scheduling priority of the process. See 5.20.

`etime` In the POSIX Locale, the elapsed time since the process was started, in the form:

[[*dd* `-`]*hh* `:`]*mm* `:`*ss*

where *dd* shall represent the number of days, *hh* the number of hours, *mm* the number of minutes, and *ss* the number of seconds. The *dd* field shall be a decimal integer. The *hh* , *mm* , and *ss* fields shall be two-digit decimal integers padded on the left with zeroes.

`time` In the POSIX Locale, the cumulative CPU time of the process in the form:

[*dd* `-`]*hh* `:`*mm* `:`*ss*

The *dd* , *hh* , *mm* , and *ss* fields shall be as described in the `etime` specifier.

`tty` The name of the controlling terminal of the process (if any) in the 7
same format used by the `who` utility. 7

`comm` The name of the command being executed (*argv* [0] value) as a string.

`args` The command with all its arguments as a string. The implementation may truncate this value to the field width; it is implementation defined whether any further truncation occurs. It is unspecified whether the string represented is a version of the argument list as it was passed to the command when it started, or is a version of the arguments as they may have been modified by the application. Applications cannot depend on being able to modify their argument list and having that modification be reflected in the output of `ps`.

Any field need not be meaningful in all implementations. In such a case a hyphen (`-`) should be output in place of the field value.

Only `comm` and `args` shall be allowed to contain `<blank>`s; all others shall not. Any implementation-defined variables shall specify in the conformance document if the field may contain `<blank>`s, as well as the default header.

Table 5-101 specifies the default header to be used in the POSIX Locale 7
corresponding to each format specifier. 7

Table 5-101 – `ps` Variable Names and Default Headers

Format Specifier	Default Header	Format Specifier	Default Header
`args`	`COMMAND`	`ppid`	`PPID`
`comm`	`COMMAND`	`rgroup`	`RGROUP`
`etime`	`ELAPSED`	`ruser`	`RUSER`
`group`	`GROUP`	`time`	`TIME`
`nice`	`NI`	`tty`	`TT`
`pcpu`	`%CPU`	`user`	`USER`
`pgid`	`PGID`	`vsz`	`VSZ`
`pid`	`PID`		

5.23.6.2 Standard Error

Used only for diagnostic messages.

5.23.6.3 Output Files

None.

5.23.7 Extended Description

None.

5.23.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.23.9 Consequences of Errors

Default.

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

Usage, Examples

The command:

```
ps -o user,pid,ppid=MOM -o args
```

writes the following in the POSIX Locale. The contents of the COMMAND field need not be the same in all implementations, due to possible truncation.

```
  USER    PID    MOM    COMMAND
helene     34     12    ps -o uid,pid,ppid=MOM -o args
```

The args format specifier is allowed to produce a truncated version of the command arguments. In some implementations, this information is no longer available when the ps utility is executed.

If the field width is too narrow to display a textual ID, the system may use a numeric version. Normally, the system would be expected to choose large enough field widths, but if a large number of fields were selected to write, it might squeeze fields to their minimum sizes to fit on one line. One way to ensure adequate width for the textual IDs is to override the default header for a field to make it larger than most or all user or group names.

There is no special quoting mechanism for header text. The header text is the rest of the argument. If multiple header changes are needed, multiple −o options can be used, such as

```
ps -o "user=User Name" -o pid=Process\ ID
```

History of Decisions Made

There is very little commonality between BSD and System V implementations of ps. Many options conflict or have subtly different usages. The working group has attempted to select a set of options that seem useful on a wide range of systems and has selected options that either can be implemented on both BSD and System V based systems without breaking the current implementations or where the options are sufficiently similar that any changes would not be unduly painful for users or implementors.

It is recognized that on some systems, especially multilevel secure systems, ps may be nearly useless. The default output has therefore been chosen both such that it does not break existing implementations and also that it is likely to provide at least some useful information on most systems.

The major change is the addition of the format specification capability. The motivation for this invention is to provide a mechanism for users to access a wider range of system information, if the system permits it, in a portable manner. The fields chosen to appear in the specification were arrived at after considering what concepts were likely to be both reasonably useful to the ''average'' user and stood a reasonable chance of being implemented on a wide range of systems. Again it is recognized that not all systems will be able to provide all the information and, conversely, some may wish to provide more. It is hoped that the approach adopted will be sufficiently flexible and extensible to accommodate most systems. Implementations may be expected to introduce new format specifiers.

The default output should consist of a short listing containing the process ID, terminal name, cumulative execution time, and command name of each process.

The preference of the group would have been to make the format specification an operand of the `ps` command. Unfortunately, BSD usage precluded this.

At one time a format was included to display the process's environment array. This was deleted because there is no portable way to display it.

The `−o nice` field is named from historical precedent. An implementation may provide an extension to name it something based on ''system scheduling priority.''

The historical `−l` option has been omitted because it could not be specified more precisely. Since the default output is ''written in an implementation-defined manner,'' having a longer version of it required is not useful in a standard. However, mentioning `−l` here in the rationale serves to reserve its meaning and implementors are encouraged to provide it as an extension.

The historical options `−g` and `−u` have had different meanings on various systems. They were changed to their uppercase equivalents to achieve a common definition in POSIX.2.

The `−A` option is equivalent to the BSD `−g` and the *SVID* `−e`. Because the two systems differed, a mnemonic compromise was selected.

The `−a` option is described with some optional behavior because the *SVID* omits session leaders, but BSD does not.

In an earlier draft, format specifiers appeared for priority and start time. The former was not defined adequately in POSIX.2 and was removed in deference to the defined system scheduling priority; the latter because elapsed time was considered to be more useful.

The following categories of information were omitted because they were too implementation-specific:

`STATE`	Process state (`R` for ''Run'',etc.)
`FLAGS`	Octal scheduling flags.
`ADDR`	Process address.
`WCHAN`	Wakeup channel or event being slept on.

In a new BSD version of `ps`, a `−O` option can be used to write all of the default information followed by additional format specifiers. This was not adopted by POSIX.2 because the default output is implementation-defined. Nevertheless, this is a useful option that should be reserved for that purpose. In the `−o` option for the POSIX.2 `ps`, the format is the concatenation of each `-o`. Therefore, the user can have an alias or function that defines the beginning of her desired format and add more fields to the end of the output in certain cases where that would be useful.

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

The `pcpu` field indicates that the CPU time available is determined in an 8
unspecified manner. This is because it is difficult to express an algorithm that is 8
useful across all possible machine architectures. Historical counterparts to this 8
value have attempted to show percentage of use in the recent past, such as the 8
preceding minute. Frequently, these values for all processed did not add up to 8
100 percent. Implementations are encouraged to provide data in this field to 8
users that will help them identify processes currently affecting the performance of 8
the system. 8