IEEE P1003.2 Draft 11.2 – September 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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

The default behavior of pax with regard to file modification times is the same as historical implementations of tar. It is not the historical behavior of cpio.

Because the −i option uses /dev/tty, utilities without a controlling terminal will not be able to use this option.

The −y option, found in earlier drafts, has been deleted because a line containing a single period for the −i option has equivalent functionality. The special lines for the −i option (a single period and the empty line) are historical practice in cpio.

In earlier drafts, an −e *charmap* option was included to increase portability of files between systems using different coded character sets. This option was omitted because it was apparent that consensus could not be formed for it. It was an interface without implementation experience and overloaded the charmap file concept to provide additional uses its original authors had not intended. The developers of POSIX.2 will consider other mechanisms for transporting files with nonportable names as they develop the new interchange format, described earlier.

The −k option was added to address international concerns about the dangers involved in the character set transformations of −e (if the target character set were different than the source, the file names might be transformed into names matching existing files) and was made more general to also protect files transferred between file systems with different {NAME_MAX} values (truncating a filename on a smaller system might also inadvertently overwrite existing files). As stated, it prevents any overwriting, even if the target file is older than the source, which is seen as a generally useful feature anyway.

It is almost certain that appropriate privileges will be required for pax to accomplish parts of this specification. Specifically, creating files of type block special or character special, restoring file access times unless the files are owned by the user (the −t option), or preserving file owner, group, and mode (the −p option) will all probably require appropriate privileges.

Some of the file characteristics referenced in this specification may not be supported by some archive formats. For example, neither the tar nor cpio formats contain the file access time. For this reason, the e specification character has been provided, intended to cause all file characteristics specified in the archive to be retained.

It is required that extracted directories, by default, have their access and modification times and permissions set to the values specified in the archive. This has obvious problems in that the directories are almost certainly modified after being extracted and that directory permissions may not permit file creation. One possible solution is to create directories with the mode specified in the archive, as modified by the *umask* of the user, plus sufficient permissions to allow file creation. After all files have been extracted, pax would then reset the access and modification times and permissions as necessary.

When the −r option is specified, and the −w option is not, implementations are permitted to overwrite files when the archive has multiple members with the same name. This may fail, of course, if permissions on the first version of the file

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

do not permit it to be overwritten.

4.49 pr — Print files

4.49.1 Synopsis

```
pr [+page] [−column] [−adFmrt] [−e[char][gap]] [−h header] [−i[char][gap]]
    [−l lines] [−n[char][width]] [−o offset] [−s[char]] [−w width] [file ...]
```

4.49.2 Description

The pr utility is a printing and pagination filter. If multiple input files are specified, each shall be read, formatted, and written to standard output. By default, the input shall be separated into 66-line pages, each with:

— A 5-line header that includes the page number, date, time, and the pathname of the file.

— A 5-line trailer consisting of blank lines.

If standard output is associated with a terminal, diagnostic messages shall be deferred until the pr utility has completed processing.

When options specifying multicolumn output are specified, output text columns shall be of equal width; input lines that do not fit into a text column shall be truncated. By default, text columns shall be separated with at least one <blank>.

4.49.3 Options

The pr utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that: the *page* option has a '+' delimiter; *page* and *column* can be multidigit numbers; some of the option-arguments are optional; and some of the option-arguments cannot be specified as separate arguments from the preceding option letter. In particular, the −s option does not allow the option letter to be separated from its argument, and the options −e, −i, and −n require that both arguments, if present, not be separated from the option letter.

The following options shall be supported by the implementation. In the following option descriptions, *column*, *lines*, *offset*, *page*, and *width* are positive decimal integers; *gap* is a nonnegative decimal integer.

+*page* Begin output at page number *page* of the formatted input.

−*column* Produce output that is *columns* wide (default shall be 1) and is written down each column in the order in which the text is received from the input file. This option should not be used with −m. The options −e and −i shall be assumed for multiple text-column output. Whether or not text columns are balanced is

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

unspecified, but a text column shall never exceed the length of the page (see the −l option). When used with −t, use the minimum number of lines to write the output.

−a Modify the effect of the -*column* option so that the columns are filled across the page in a round-robin order (e.g., when *column* is 2, the first input line heads column 1, the second heads column 2, the third is the second line in column 1, etc.).

−d Produce output that is double-spaced; append an extra <newline> following every <newline> found in the input.

−e[*char*][*gap*]
Expand each input <tab> to the next greater column position specified by the formula *n*∗*gap* +1, where *n* is an integer > 0. If *gap* is zero or is omitted, it shall default to 8. All <tab> characters in the input shall be expanded into the appropriate number of <space>s. If any nondigit character, *char* , is specified, it shall be used as the input tab character.

−F Use a <form-feed> character for new pages, instead of the default behavior that uses a sequence of <newline> characters.

−h *header* Use the string *header* to replace the contents of the *file* operand in the page header. See 4.49.6.1.

−i[*char*][*gap*]
In output, replace multiple <space>s with <tab>s wherever two or more adjacent <space>s reach column positions *gap* +1, 2∗*gap* +1, 3∗*gap* +1, etc. If *gap* is zero or is omitted, default <tab> settings at every eighth column position shall be assumed. If any nondigit character, *char* , is specified, it shall be used as the output <tab> character.

−l *lines* Override the 66-line default and reset the page length to *lines* . If *lines* is not greater than the sum of both the header and trailer depths (in lines), the pr utility shall suppress both the header and trailer, as if the −t option were in effect.

−m Merge files. Standard output shall be formatted so the pr utility writes one line from each file specified by a *file* operand, side by side into text columns of equal fixed widths, in terms of the number of column positions. Implementations shall support merging of at least nine *file* operands.

−n[*char*][*width*]
Provide *width* -digit line numbering (default for *width* shall be 5). The number shall occupy the first *width* column positions of each text column of default output or each line of −m output. If *char* (any nondigit character) is given, it shall be appended to the line number to separate it from whatever follows (default for *char* shall be a <tab>).

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

10480 `-o` *offset* Each line of output shall be preceded by offset <space>s. If the `-o`
10481 option is not specified, the default offset shall be zero. The space
10482 taken shall be in addition to the output line width (see `-w` option
10483 below).

10484 `-r` Write no diagnostic reports on failure to open files.

10485 `-s`[*char*] Separate text columns by the single character *char* instead of by
10486 the appropriate number of <space>s (default for *char* shall be
10487 the <tab> character).

10488 `-t` Write neither the five-line identifying header nor the five-line
10489 trailer usually supplied for each page. Quit writing after the last
10490 line of each file without spacing to the end of the page.

10491 `-w` *width* Set the width of the line to *width* column positions for multiple
10492 text-column output only. If the `-w` option is not specified and the
10493 `-s` option is not specified, the default width shall be 72. If the `-w`
10494 option is not specified and the `-s` option is specified, the default
10495 width shall be 512.

10496 For single column output, input lines shall not be truncated.

10497 ### 4.49.4 Operands

10498 The following operand shall be supported by the implementation:

10499 *file* A pathname of a file to be written. If no *file* operands are
10500 specified, or if a *file* operand is –, the standard input shall be
10501 used.

10502 ### 4.49.5 External Influences

10503 #### 4.49.5.1 Standard Input

10504 The standard input shall be used only if no *file* operands are specified, or if a *file*
10505 operand is –. See Input Files.

10506 #### 4.49.5.2 Input Files

10507 The input files shall be text files.

10508 #### 4.49.5.3 Environment Variables

10509 The following environment variables shall affect the execution of `pr`:

10510 **LANG** This variable shall determine the locale to use for the
10511 locale categories when both **LC_ALL** and the correspond-
10512 ing environment variable (beginning with **LC_**) do not
10513 specify a locale. See 2.6.

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

10514 **LC_ALL** This variable shall determine the locale to be used to over-
10515 ride any values for locale categories specified by the set-
10516 tings of **LANG** or any environment variables beginning
10517 with **LC_**.

10518 **LC_CTYPE** This variable shall determine the locale for the interpreta-
10519 tion of sequences of bytes of text data as characters (e.g.,
10520 single- versus multibyte characters in arguments and
10521 input files) and which characters are defined as printable
10522 (character class `print`). Nonprintable characters still
10523 shall be written to standard output, but shall be not
10524 counted for the purpose for column-width and line-length
10525 calculations.

10526 **LC_MESSAGES** This variable shall determine the language in which mes-
10527 sages should be written.

10528 **LC_TIME** This variable shall determine the format of the date and
10529 time for use in writing header lines.

10530 **TZ** This variable shall determine the time zone for use in
10531 writing header lines.

10532 #### 4.49.5.4 Asynchronous Events

10533 If `pr` receives an interrupt while writing to a terminal, it shall flush all accumu-
10534 lated error messages to the screen before terminating.

10535 ### 4.49.6 External Effects

10536 #### 4.49.6.1 Standard Output

10537 The `pr` utility output shall be a paginated version of the original file (or files).
10538 This pagination shall be accomplished using either `<form-feed>`s or a sequence
10539 of `<newline>`s, as controlled by the `-F` option. Page headers shall be generated
10540 unless the `-t` option is specified. The page headers shall be of the form:

10541 `"\n\n%s %s Page %d\n\n\n"`, *<output of* `date`*>*, *<file>*,
10542 *<page number>*

10543 In the POSIX Locale, the *<output of* `date`*>* field, representing the date and time of
10544 last modification of the input file (or the current date and time if the input file is
10545 standard input), shall be equivalent to the output of the following command as it
10546 would appear if executed at the given time:

10547 `date "+%b %e %H:%M %Y"`

10548 without the trailing `<newline>`, if the page being written is from standard input.
10549 If the page being written is not from standard input, in the POSIX Locale, the
10550 same format shall be used, but the time used shall be the modification time of the
10551 file corresponding to *file* instead of the current time. When the LC_TIME locale

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

category is not set to the POSIX Locale, a different format and order of presentation of this field may be used.

If the standard input is used instead of a *file* operand, the <*file*> field shall be replaced by a null string.

If the –h option is specified, the *file* field shall be replaced by the *header* argument.

4.49.6.2 Standard Error

Used only for diagnostic messages.

4.49.6.3 Output Files

None.

4.49.7 Extended Description

None.

4.49.8 Exit Status

The pr utility shall exit with one of the following values:

0 All files were written successfully.

>0 An error occurred.

4.49.9 Consequences of Errors

Default.

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

Examples, Usage

To print a numbered list of all files in the current directory:

```
ls -a | pr -n -h "Files in $(pwd)."
```

History of Decisions Made

This utility is one of those that does not follow the Utility Syntax Guidelines because of its historical origins. The working group could have added new options that obeyed the guidelines (and marked the old options *obsolescent*) or devised an entirely new utility; there are examples of both actions in this standard. For this utility, it chose to leave some of the options as they are because of their heavy usage by existing applications. However, due to interest in the international

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

10581 community, the developers of the standard have agreed to provide an alternative
10582 syntax for the next version of this standard that conforms to the spirit of the Util-
10583 ity Syntax Guidelines. This new syntax will be accompanied by the existing syn-
10584 tax, marked as obsolescent. System implementors are encouraged to develop and
10585 promulgate a new syntax for `pr`, perhaps using a different utility name, that can
10586 be adopted for the next version of this standard.

10587 Implementations are required to accept option arguments to the `-h`, `-l`, `-o`, and
10588 `-w` options whether presented as part of the same argument or as a separate argu-
10589 ment to `pr`, as suggested by the utility syntax guidelines. The `-n` and `-s` options,
10590 however, are specified as in historical practice because they are frequently
10591 specified without their optional arguments. If a `<blank>` were allowed before the
10592 option-argument in these cases, a file operand could mistakenly be interpreted as
10593 an option-argument in historical applications.

10594 Historical implementations of the `pr` utility have differed in the action taken for
10595 the `-f` option. BSD uses it as described here for the `-F` option; System V uses it to
10596 change trailing `<newline>`s on each page to a `<form-feed>` and, if standard
10597 output is a TTY device, sends an `<alert>` to standard error and reads a line from
10598 `/dev/tty` before the first page. Draft 9 incorrectly specified part of the System V
10599 behavior, raising several ballot objections. There were strong arguments from
10600 both sides of this issue concerning existing practice and additional arguments
10601 against the System V `-f` behavior, on the grounds that it was not a modular
10602 design to have the behavior of an option change depending on where output is
10603 directed. Therefore, the `-f` option is not specified and the `-F` option has been
10604 added.

10605 The `-p` option was omitted since it represents a purely interactive usage. 1

10606 The *<output of date>* field in the `-l` format is specified only for the POSIX Locale.
10607 As noted, the format can be different in other locales. No mechanism for defining
10608 this is present in this standard, as the appropriate vehicle is a messaging system;
10609 i.e., the format should be specified as a ‘‘message.’’

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

10610 ## 4.50 `printf` — Write formatted output

10611 ### 4.50.1 Synopsis

10612 `printf` *format* [*argument* ...]

10613 ### 4.50.2 Description

10614 The `printf` utility shall write formatted operands to the standard output. The
10615 *argument* operands shall be formatted under control of the *format* operand.

10616 ### 4.50.3 Options

10617 None.

10618 ### 4.50.4 Operands

10619 The following operands shall be supported by the implementation:

10620 *format* A string describing the format to use to write the remaining
10621 operands; see 4.50.7.

10622 *argument* The strings to be written to standard output, under the control of
10623 *format* ; see 4.50.7.

10624 ### 4.50.5 External Influences

10625 #### 4.50.5.1 Standard Input

10626 None.

10627 #### 4.50.5.2 Input Files

10628 None.

10629 #### 4.50.5.3 Environment Variables

10630 The following environment variables shall affect the execution of `printf`:

10631 **LANG** This variable shall determine the locale to use for the
10632 locale categories when both **LC_ALL** and the correspond-
10633 ing environment variable (beginning with **LC_**) do not
10634 specify a locale. See 2.6.

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

10635 **LC_ALL** This variable shall determine the locale to be used to over-
10636 ride any values for locale categories specified by the set-
10637 tings of **LANG** or any environment variables beginning
10638 with **LC_**.

10639 **LC_CTYPE** This variable shall determine the locale for the interpreta-
10640 tion of sequences of bytes of text data as characters (e.g.,
10641 single- versus multibyte characters in arguments).

10642 **LC_MESSAGES** This variable shall determine the language in which mes-
10643 sages should be written.

10644 **LC_NUMERIC** This variable shall determine the locale for numeric for-
10645 matting. It shall affect the format of numbers written
10646 using the `e`, `E`, `f`, `g`, and `G` conversion characters (if sup-
10647 ported).

10648 #### 4.50.5.4 Asynchronous Events

10649 Default.

10650 ### 4.50.6 External Effects

10651 #### 4.50.6.1 Standard Output

10652 See 4.50.7.

10653 #### 4.50.6.2 Standard Error

10654 Used only for diagnostic messages.

10655 #### 4.50.6.3 Output Files

10656 None.

10657 ### 4.50.7 Extended Description

10658 The *format* operand shall be used as the *format* string described in 2.12 with the
10659 following exceptions:

10660 (1) A `<space>` character in the format string, in any context other than a
10661 flag of a conversion specification, shall be treated as an ordinary charac-
10662 ter that is copied to the output.

10663 (2) A Δ character in the format string shall be treated as a Δ character, not as
10664 a `<space>`.

10665 (3) In addition to the escape sequences shown in Table 2-15 (see 2.12), *ddd* ,
10666 where *ddd* is a one-, two-, or three-digit octal number, shall be written as
10667 a byte with the numeric value specified by the octal number.

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

(4) The implementation shall not precede or follow output from the `d` or `u` conversion specifications with `<blank>`s not specified by the *format* operand.

(5) The implementation shall not precede output from the `o` conversion specification with zeroes not specified by the *format* operand.

(6) The `e`, `E`, `f`, `g`, and `G` conversion specifications need not be supported.

(7) An additional conversion character, `b`, shall be supported as follows. The argument shall be taken to be a string that may contain backslash-escape sequences. The following backslash-escape sequences shall be supported:

(a) The escape sequences listed in Table 2-15, which shall be converted to the characters they represent;

(b) `\0`*ddd* , where *ddd* is a zero-, one-, two-, or three-digit octal number that shall be converted to a byte with the numeric value specified by the octal number;

(c) `\c`, which shall not be written and shall cause `printf` to ignore any remaining characters in the string operand containing it, any remaining string operands, and any additional characters in the *format* operand.

The interpretation of a backslash followed by any other sequence of characters is unspecified.

Bytes from the converted string shall be written until the end of the string or the number of bytes indicated by the precision specification is reached. If the precision is omitted, it shall be taken to be infinite, so all bytes up to the end of the converted string shall be written.

(8) For each specification that consumes an argument, the next argument operand shall be evaluated and converted to the appropriate type for the conversion as specified below.

(9) The *format* operand shall be reused as often as necessary to satisfy the argument operands. Any extra `c` or `s` conversion specifications shall be evaluated as if a null string argument were supplied; other extra conversion specifications shall be evaluated as if a zero argument were supplied. If the *format* operand contains no conversion specifications and *argument* operands are present, the results are unspecified.

(10) If a character sequence in the *format* operand begins with a `%` character, but does not form a valid conversion specification, the behavior is unspecified.

The *argument* operands shall be treated as strings if the corresponding conversion character is `b`, `c`, or `s`; otherwise, it shall be evaluated as a C constant, as described by the C Standard {7},with the following extensions:

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

— A leading plus or minus sign shall be allowed.

— If the leading character is a single- or double-quote, the value shall be the numeric value in the underlying code set of the character following the single- or double-quote.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message shall be written to standard error and the utility shall not exit with a zero exit status, but shall continue processing any remaining operands and shall write the value accumulated at the time the error was detected to standard output.

4.50.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

4.50.9 Consequences of Errors

Default.

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

Examples, Usage

To alert the user and then print and read a series of prompts:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone number: "
read phone
```

To read out a list of right and wrong answers from a file, calculate the percentage right, and print them out. The numbers are right-justified and separated by a single <tab>. The percentage is written to one decimal place of accuracy.

```
while read right wrong ; do
   percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
   printf "%2d right\t%2d wrong\t(%s%%)\n" \
           $right $wrong $percent
done < database_file
```

The command:

```
printf "%5d%4d\n" 1 21 321 4321 54321
```

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

produces:

```
          1  21
        3214321
      54321   0
```

Note that the *format* operand is used three times to print all of the given strings and that a `0` was supplied by `printf` to satisfy the last `%4d` conversion specification.

The `printf` utility is required to notify the user when conversion errors are detected while producing numeric output; thus, the following results would be expected on an implementation with 32-bit twos-complement integers when `%d` is specified as the *format* operand:

Argument	Standard Output	Diagnostic Output
`5a`	`5`	`printf: "5a" not completely converted`
`9999999999`	`2147483647`	`printf: "9999999999" arithmetic overflow`
`-9999999999`	`-2147483648`	`printf: "-9999999999" arithmetic overflow`
`ABC`	`0`	`printf: "ABC" expected numeric value`

The diagnostic message format is not specified, but these examples convey the type of information that should be reported. Note that the value shown on standard output is what would be expected as the return value from the C Standard {7} function *strtol* (). A similar correspondence exists between `%u` and *strtoul* () and `%e`, `%f`, and `%g` (if the implementation supports floating-point conversions) and *strtod* ().

In a locale using ISO/IEC 646 {1} as the underlying code set, the command:

```
printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"
```

produces:

`3` Numeric value of constant 3
`3` Numeric value of constant 3
`−3` Numeric value of constant −3
`51` Numeric value of the character ‘‘3’’ in ISO/IEC 646 {1} code set
`43` Numeric value of the character ‘‘+’’ in ISO/IEC 646 {1} code set
`45` Numeric value of the character ‘‘-’’ in ISO/IEC 646 {1} code set

Note that in a locale with multibyte characters, the value of a character is intended to be the value of the equivalent of the *wchar_t* representation of the character as described in C Standard {7}.

History of Decisions Made

The `printf` utility was added to provide functionality that has historically been provided by `echo`. However, due to irreconcilable differences in the various versions of `echo` extant, the version in this standard has few special features, leaving those to this new `printf` utility, which is based on one in the Ninth Edition at AT&T Bell Labs.

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

The Extended Description almost exactly matches the C Standard {7} *printf* () function, although it is described in terms of the file format notation in 2.12.

The floating point formatting conversion specifications are not required because all arithmetic in the shell is integer arithmetic. The `awk` utility performs floating point calculations and provides its own `printf` function. The `bc` utility can perform arbitrary-precision floating point arithmetic, but doesn't provide extensive formatting capabilities. (This `printf` utility cannot really be used to format `bc` output; it does not support arbitrary precision.) Implementations are encouraged to support the floating point conversions as an extension.

Note that this `printf` utility, like the C Standard {7} *printf* () function on which it is based, makes no special provision for dealing with multibyte characters when using the `%c` conversion specification or when a precision is specified in a `%b` or `%s` conversion specification. Applications should be extremely cautious using either of these features when there are multibyte characters in the character set.

Field widths and precisions cannot be specified as `'*'` since the `'*'` can be replaced directly in the *format* operand using shell variable substitution. Implementations can also provide this feature as an extension if they so choose.

Hexadecimal character constants as defined in the C Standard {7} are not recognized in the *format* operand because there is no consistent way to detect the end of the constant. Octal character constants are limited to, at most, three octal digits, but hexadecimal character constants are only terminated by a nonhex-digit character. In the C Standard {7}, the ## concatenation operator can be used to terminate a constant and follow it with a hexadecimal character to be written. In the shell, concatenation occurs before the `printf` utility has a chance to parse the end of the hexadecimal constant.

The `%b` conversion specification is not part of the C Standard {7}; it has been added here as a portable way to process backslash-escapes expanded in string operands as provided by the System V version of the `echo` utility. See also the rationale for `echo` for ways to use `printf` as a replacement for all of the traditional versions of the `echo` utility.

If an argument cannot be parsed correctly for the corresponding conversion specification, the `printf` utility is required to report an error. Thus, overflow and extraneous characters at the end of an argument being used for a numeric conversion are to be reported as errors. If written in C, the `printf` utility could use the *strtol* () function to parse optionally signed numeric arguments, *strtoul* () to parse unsigned numeric arguments, and *strtod* () to parse floating point arguments (if floating point conversions are supported). It is not considered an error if an argument operand is not completely used for a `c` or `s` conversion or if a ''string'' operand's first or second character is used to get the numeric value of a character.

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

10820 ## 4.51 pwd — Return working directory name

10821 ### 4.51.1 Synopsis

10822 pwd

10823 ### 4.51.2 Description

10824 The pwd utility shall write an absolute pathname of the current working directory
10825 to standard output.

10826 ### 4.51.3 Options

10827 None.

10828 ### 4.51.4 Operands

10829 None.

10830 ### 4.51.5 External Influences

10831 #### 4.51.5.1 Standard Input

10832 None.

10833 #### 4.51.5.2 Input Files

10834 None.

10835 #### 4.51.5.3 Environment Variables

10836 The following environment variables shall affect the execution of pwd:

10837 **LANG** This variable shall determine the locale to use for the
10838 locale categories when both **LC_ALL** and the correspond-
10839 ing environment variable (beginning with **LC_**) do not
10840 specify a locale. See 2.6.

10841 **LC_ALL** This variable shall determine the locale to be used to over-
10842 ride any values for locale categories specified by the set-
10843 tings of **LANG** or any environment variables beginning
10844 with **LC_**.

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

10845 **LC_MESSAGES** This variable shall determine the language in which mes-
10846 sages should be written.

10847 #### 4.51.5.4 Asynchronous Events

10848 Default.

10849 ### 4.51.6 External Effects

10850 #### 4.51.6.1 Standard Output

10851 The `pwd` utility output shall be an absolute pathname of the current working
10852 directory:

10853 `"%s\n"`, *<directory pathname>*

10854 #### 4.51.6.2 Standard Error

10855 Used only for diagnostic messages.

10856 #### 4.51.6.3 Output Files

10857 None.

10858 ### 4.51.7 Extended Description

10859 None.

10860 ### 4.51.8 Exit Status

10861 The `pwd` utility shall exit with one of the following values:

10862 0 Successful completion.

10863 >0 An error occurred.

10864 ### 4.51.9 Consequences of Errors

10865 If an error is detected, output shall not be written to standard output, a diagnos-
10866 tic message shall be written to standard error, and the exit status shall not be
10867 zero.

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

10868 **4.51.10 Rationale.** *(This subclause is not a part of P1003.2)*

10869 **Examples, Usage**

10870 Some implementations have historically provided `pwd` as a shell special built-in
10871 command.

10872 **History of Decisions Made**

10873 In most utilities, if an error occurs, partial output may be written to standard out-
10874 put. This does not happen in historical implementations of `pwd`. Because `pwd` is
10875 frequently used in existing shell scripts without checking the exit status, it is
10876 important that the historical behavior is required here; therefore, the Conse-
10877 quences of Errors subclause specifically disallows any partial output being writ-
10878 ten to standard output.

10879 ## 4.52 `read` — Read a line from standard input

10880 ### 4.52.1 Synopsis

10881 `read [-r]` *var* ...

10882 ### 4.52.2 Description

10883 The `read` utility shall read a single line from standard input.

10884 By default, unless the `-r` option is specified, backslash (\) shall act as an escape
10885 character, as described in 3.2.1.

10886 The line shall be split into fields (see the definition in 3.1.3) as in the shell (see
10887 3.6.5); the first field shall be assigned to the first variable *var,* the second field to
10888 the second variable *var,* etc. If there are fewer *var* operands specified than there
10889 are fields, the leftover fields and their intervening separators shall be assigned to
10890 the last *var*. If there are fewer fields than *var*s, the remaining *var*s shall be set to
10891 empty strings.

10892 The setting of variables specified by the *var* operands shall affect the current shell
10893 execution environment; see 3.12.

10894 ### 4.52.3 Options

10895 The `read` utility shall conform to the utility argument syntax guidelines
10896 described in 2.10.2.

10897 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.

10898 `-r` Do not treat a backslash character in any special way. Consider
10899 each backslash to be part of the input line.

10900 ### 4.52.4 Operands

10901 The following operands shall be supported by the implementation:

10902 *var* The name of an existing or nonexisting shell variable.

10903 ### 4.52.5 External Influences

10904 #### 4.52.5.1 Standard Input

10905 The standard input shall be a text file.

10906 #### 4.52.5.2 Input Files

10907 None.

10908 #### 4.52.5.3 Environment Variables

10909 The following environment variables shall affect the execution of `read`:

10910 **IFS** This variable shall determine the internal field separators
10911 used to delimit fields. See 3.5.3.

10912 **LANG** This variable shall determine the locale to use for the
10913 locale categories when both **LC_ALL** and the correspond-
10914 ing environment variable (beginning with **LC_**) do not
10915 specify a locale. See 2.6.

10916 **LC_ALL** This variable shall determine the locale to be used to over-
10917 ride any values for locale categories specified by the set-
10918 tings of **LANG** or any environment variables beginning
10919 with **LC_**. 2.6.

10920 **LC_CTYPE** This variable shall determine the locale for the interpreta-
10921 tion of sequences of bytes of text data as characters (e.g.,
10922 single- versus multibyte characters in arguments).

10923 **LC_MESSAGES** This variable shall determine the language in which mes-
10924 sages should be written.

10925 #### 4.52.5.4 Asynchronous Events

10926 Default.

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

10927 ### 4.52.6 External Effects

10928 #### 4.52.6.1 Standard Output

10929 None.

10930 #### 4.52.6.2 Standard Error

10931 Used only for diagnostic messages.

10932 #### 4.52.6.3 Output Files

10933 None.

10934 ### 4.52.7 Extended Description

10935 None.

10936 ### 4.52.8 Exit Status

10937 The `read` utility shall exit with one of the following values:

10938 0 Successful completion.

10939 >0 End-of-file was detected or an error occurred.

10940 ### 4.52.9 Consequences of Errors

10941 Default.

10942 ### 4.52.10 Rationale. *(This subclause is not a part of P1003.2)*

10943 **Examples, Usage**

10944 The following command:

```
10945       while read -r xx yy
10946       do
10947               printf "%s %s\n" "$yy" "$xx"                    1
10948       done < input_file
```

10949 prints a file with the first field of each line moved to the end of the line.

10950 The text in 2.11.5.2 indicates that the results are undefined if an end-of-file is
10951 detected following a backslash at the end of a line when `-r` is not specified.

10952 Since `read` affects the current shell execution environment, it is generally pro-
10953 vided as a shell regular built-in. If it is called in a subshell or separate utility 1
10954 execution environment, such as one of the following: 1

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

```
(read foo)
nohup read ...
find . -exec read ... \;
```

it will not affect the shell variables in the caller's environment.

History of Decisions Made

The `read` utility has historically been a shell built-in. It was separated off into its own clause to take advantage of the standard's richer description of functionality at the utility level.

The `-r` option was added to enable `read` to subsume the purpose of the historical `line` utility.

4.53 `rm` — Remove directory entries

4.53.1 Synopsis

`rm [-fiRr]` *file* ...

4.53.2 Description

The `rm` utility shall remove the directory entry specified by each *file* argument.

If either of the files dot or dot-dot are specified as the basename portion of an operand (i.e., the final pathname component), `rm` shall write a diagnostic message to standard error and do nothing more with such operands.

For each *file* the following steps shall be taken:

(1) If the *file* does not exist:

  (a) If the `-f` option is not specified, write a diagnostic message to standard error.

  (b) Go on to any remaining *files*.

(2) If *file* is of type directory, the following steps shall be taken:

  (a) If neither the `-R` option nor the `-r` option is specified, write a diagnostic message to standard error, do nothing more with *file*, and go on to any remaining files.

  (b) If the `-f` option is not specified, and either the permissions of *file* do not permit writing and the standard input is a terminal or the `-i` option is specified, write a prompt to standard error and read a line from the standard input. If the response is not affirmative, do nothing more with the current file and go on to any remaining files.

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

(c) For each entry contained in *file*, other than dot or dot-dot, the four steps listed here [(1)-(4)] shall be taken with the entry as if it were a *file* operand.

(d) If the −i option is specified, write a prompt to standard error and read a line from the standard input. If the response is not affirmative, do nothing more with the current file, and go on to any remaining files.

(3) If *file* is not of type directory, the −f option is not specified, and either the permissions of *file* do not permit writing and the standard input is a terminal or the −i option is specified, write a prompt to the standard error and read a line from the standard input. If the response is not affirmative, do nothing more with the current file and go on to any remaining files.

(4) If the current file is a directory, rm shall perform actions equivalent to the POSIX.1 {8} *rmdir* () function called with a pathname of the current file used as the *path* argument. If the current file is not a directory, rm shall perform actions equivalent to the POSIX.1 {8} *unlink* () function called with a pathname of the current file used as the *path* argument.

If this fails for any reason, rm shall write a diagnostic message to standard error, do nothing more with the current file, and go on to any remaining files.

The rm utility shall be able to descend to arbitrary depths in a file hierarchy, and shall not fail due to path length limitations (unless an operand specified by the user exceeds system limitations).

4.53.3 Options

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

The following options shall be supported by the implementation:

−f Do not prompt for confirmation. Do not write diagnostic messages or modify the exit status in the case of nonexistent operands. Any previous occurrences of the −i option shall be ignored.

−i Prompt for confirmation as described in 4.53.2. Any previous occurrences of the −f option shall be ignored.

−R Remove file hierarchies. See 4.53.2.

−r Equivalent to −R.

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