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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

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.