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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

```
(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.

11023 ### 4.53.4 Operands

11024 The following operand shall be supported by the implementation:

11025 *file* A pathname of a directory entry to be removed.

11026 ### 4.53.5 External Influences

11027 #### 4.53.5.1 Standard Input

11028 Used to read an input line in response to each prompt specified in 4.53.6.1. Oth-
11029 erwise, the standard input shall not be used.

11030 #### 4.53.5.2 Input Files

11031 None.

11032 #### 4.53.5.3 Environment Variables

11033 The following environment variables shall affect the execution of `rm`:

11034 **LANG** This variable shall determine the locale to use for the
11035 locale categories when both **LC_ALL** and the correspond-
11036 ing environment variable (beginning with **LC_**) do not
11037 specify a locale. See 2.6.

11038 **LC_ALL** This variable shall determine the locale to be used to over-
11039 ride any values for locale categories specified by the set-
11040 tings of **LANG** or any environment variables beginning
11041 with **LC_**.

11042 **LC_COLLATE** This variable shall determine the locale for the behavior of
11043 ranges, equivalence classes, and multicharacter collating
11044 elements used in the extended regular expression defined
11045 for the `yesexpr` locale keyword in the LC_MESSAGES
11046 category.

11047 **LC_CTYPE** This variable shall determine the locale for the interpreta-
11048 tion of sequences of bytes of text data as characters (e.g.,
11049 single- versus multibyte characters in arguments) and the
11050 behavior of character classes within regular expressions
11051 used in the extended regular expression defined for the
11052 `yesexpr` locale keyword in the LC_MESSAGES category.

11053 **LC_MESSAGES** This variable shall determine the processing of affirmative
11054 responses and the language in which messages should be
11055 written.

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

11056 #### 4.53.5.4 Asynchronous Events

11057 Default.

11058 ### 4.53.6 External Effects

11059 #### 4.53.6.1 Standard Output

11060 None.

11061 #### 4.53.6.2 Standard Error

11062 Prompts shall be written to standard error under the conditions specified in
11063 4.53.2 and 4.53.3. The prompts shall contain the *file* pathname, but their format
11064 is otherwise unspecified. The standard error shall also be used for diagnostic
11065 messages.

11066 #### 4.53.6.3 Output Files

11067 None.

11068 ### 4.53.7 Extended Description

11069 None.

11070 ### 4.53.8 Exit Status

11071 The `rm` utility shall exit with one of the following values:

11072 0 If the `-f` option was not specified, all the named directory entries were
11073 removed; otherwise, all the existing named directory entries were
11074 removed.

11075 >0 An error occurred.

11076 ### 4.53.9 Consequences of Errors

11077 Default.

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

11078 **4.53.10 Rationale.** *(This subclause is not a part of P1003.2)*

11079 **Examples, Usage**

11080 The *SVID* requires that systems do not permit the removal of the last link to an
11081 executable binary file that is being executed. Thus, the `rm` utility can fail to
11082 remove such files.

11083 The `-i` option causes `rm` to prompt and read the standard input even if the stan-
11084 dard input is not a terminal, but in the absence of `-i` the mode prompting is not
11085 done when the standard input is not a terminal.

11086 For absolute clarity, paragraphs (2)(b) and (3) in 4.53.2, describing `rm`'s behavior
11087 when prompting for confirmation, should be interpreted in the following manner:

```
11088     if ((NOT f_option) AND
11089             ((not_writable AND input_is_terminal) OR i_option))
```

11090 It is forbidden to remove the names dot and dot-dot in order to avoid the conse-
11091 quences of inadvertently doing something like:

```
11092     rm -r .*
```

11093 The following command

```
11094     rm a.out core
```

11095 removes the directory entries `a.out` and `core`.

11096 The following command

```
11097     rm -Rf junk
```

11098 removes the directory `junk` and all its contents, without prompting.

11099 **History of Decisions Made**

11100 The exact format of the interactive prompts is unspecified. Only the general
11101 nature of the contents of prompts are specified, because implementations may
11102 desire more descriptive prompts than those used on historical implementations.
11103 Therefore, an application not using the `-f` option, or using the `-i` option relies on
11104 the system to provide the most suitable dialogue directly with the user, based on
11105 the behavior specified.

11106 The `-r` option is existing practice on all known systems. The synonym `-R` option
11107 is provided for consistency with the other utilities in this standard that provide
11108 options requesting recursive descent.

11109 The behavior of the `-f` option in historical versions of `rm` is inconsistent. In gen-
11110 eral, along with ''forcing'' the unlink without prompting for permission, it always
11111 causes diagnostic messages to be suppressed and the exit status to be unmodified
11112 for nonexistent operands and files that cannot be unlinked. In some versions,
11113 however, the `-f` option suppresses usage messages and system errors as well.
11114 Suppressing such messages is not a service to either shell scripts or users.

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

It is less clear that error messages regarding unlinkable files should be suppressed. Although this is historical practice, this standard does not permit the `-f` option to suppress such messages.

When given the `-r` and `-i` options, historical versions of `rm` prompt the user twice for each directory, once before removing its contents and once before actually attempting to delete the directory entry that names it. This allows the user to ''prune'' the file hierarchy walk. Historical versions of `rm` were inconsistent in that some did not do the former prompt for directories named on the command line and others had obscure prompting behavior when the `-i` option was specified and the permissions of the file did not permit writing. The POSIX.2 `rm` differs little from historic practice, but does require that prompts be consistent. Historical versions of `rm` were also inconsistent in that prompts were done to both standard output and standard error. POSIX.2 requires that prompts be done to standard error, for consistency with `cp` and `mv` and to allow existing extensions to `rm` that provide an option to list deleted files on standard output.

The `rm` utility is required to descend to arbitrary depths so that any file hierarchy may be deleted. This means, for example, that the `rm` utility cannot run out of file descriptors during its descent, i.e., if the number of file descriptors is limited, `rm` cannot be implemented in the historical fashion where a file descriptor is used per directory level. Also, `rm` is not permitted to fail because of path length restrictions, unless an operand specified by the user is longer than {PATH_MAX}.

## 4.54 `rmdir` — Remove directories

### 4.54.1 Synopsis

`rmdir [-p]` *dir* ...

### 4.54.2 Description

The `rmdir` utility shall remove the directory entry specified by each *dir* operand, which shall refer to an empty directory.

Directories shall be processed in the order specified. If a directory and a subdirectory of that directory are specified in a single invocation of the `rmdir` utility, the subdirectory shall be specified before the parent directory so that the parent directory will be empty when the `rmdir` utility tries to remove it.

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