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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

print out together.

The −c option may imply a spooling operation, but this is not required. The utility can be implemented to simply wait until the printer is ready and then wait until it's finished. Because of that, there is no attempt to define a queueing mechanism (priorities, classes of output, etc.).

The −n and −d options were added in response to balloting objections that too little historical value was being provided.

Although the historical System V lp and BSD lpr utilities have provided similar functionality, they used different names for the environment variable specifying the destination printer. Since the name of the utility here is lp, **LPDEST** (used by the System V lp utility) was given precedence over **PRINTER** (used by the BSD lpr utility). Since environments of users frequently contain one or the other environment variable, the lp utility is required to recognize both. If this was not done, many applications would send output to unexpected output devices when users moved from system to system.

Some have commented that lp has far too little functionality to make it worthwhile. Requests have proposed additional options or operands or both that added functionality. The requests included:

— wording *requiring* the output to be ''hardcopy''

— a requirement for multiple printers

— options for PostScript, dimpress, hp, and lineprint formats

Given that a POSIX.2 compliant system is not required to even have a printer, placing further restrictions upon the behavior of the printer is not useful. Since hardcopy format is so application dependent, it is difficult, if not impossible, to select a reasonable subset of functionality that should be required on all POSIX.2 compliant systems.

The term ''unspecified'' is used in this clause in lieu of ''implementation defined'' as most known implementations would not be able to say anything fully useful in their conformance documents: the existence and usage of printers is very dependent on how the system administrator configures each individual system.

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

4.39 `ls` — List directory contents

4.39.1 Synopsis

`ls [-CFRacdilqrtu1][`*file* `...]`

4.39.2 Description

For each operand that names a file of a type other than directory, `ls` shall write the name of the file as well as any requested, associated information. For each operand that names a file of type directory, `ls` shall write the names of files contained within that directory, as well as any requested, associated information.

If no operands are specified, the contents of the current directory shall be written. If more than one operand is specified, nondirectory operands shall be written first; directory and nondirectory operands shall be sorted separately according to the collating sequence in the current locale.

4.39.3 Options

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

The following options shall be supported by the implementation:

`-C` Write multi-text-column output with entries sorted down the columns, according to the collating sequence. The number of text columns and the column separator characters are unspecified, but should be adapted to the nature of the output device.

`-F` Write a slash (`/`) immediately after each pathname that is a directory, an asterisk (`*`) after each that is executable, and a vertical bar (`|`) after each that is a FIFO.

`-R` Recursively list subdirectories encountered.

`-a` Write out all directory entries, including those whose names begin with a period (`.`). Entries beginning with a period (.) shall not be written out unless explicitly referenced, the `-a` option is supplied, or an implementation-defined condition causes them to be written.

`-c` Use time of last modification of the file status information (see POSIX .1 {8}5.6.1.3) instead of last modification of the file itself for sorting (`-t`) or writing (`-l`).

`-d` Do not treat directories differently than other types of files. The use of `-d` with `-R` produces unspecified results. 2 2

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

8322 `-i` For each file, write the file's file serial number (see POSIX.1 {8}
8323 5.6.2).

8324 `-l` (The letter ell.) Write out in long format (see 4.39.6.1). When `-l` 2
8325 (ell) is specified, `-1` (one) shall be assumed. 2

8326 `-q` Force each instance of nonprintable filename characters and 2
8327 `<tab>`s to be written as the question-mark (`?`) character. Imple-
8328 mentations may provide this option by default if the output is to a
8329 terminal device.

8330 `-r` Reverse the order of the sort to get reverse collating sequence or
8331 oldest first.

8332 `-t` Sort by time modified (most recently modified first) before sorting
8333 the operands by the collating sequence.

8334 `-u` Use time of last access (see POSIX.1 {8} 5.6.1.3) instead of last
8335 modification of the file for sorting (`-t`) or writing (`-l`).

8336 `-1` (The numeric digit one.) Force output to be one entry per line.

8337 Specifying more than one of the options in the following mutually exclusive pairs 2
8338 shall not be considered an error: `-C` and `-l` (ell), `-C` and `-1` (one), `-c` and `-u`. The 2
8339 last option specified in each pair shall determine the output format. 2

8340 ### 4.39.4 Operands

8341 The following operands shall be supported by the implementation:

8342 *file* A pathname of a file to be written. If the file specified is not
8343 found, a diagnostic message shall be output on standard error.

8344 ### 4.39.5 External Influences

8345 #### 4.39.5.1 Standard Input

8346 None.

8347 #### 4.39.5.2 Input Files

8348 None.

8349 #### 4.39.5.3 Environment Variables

8350 The following environment variables shall affect the execution of `ls`:

8351 **COLUMNS** This variable shall determine the user's preferred column
8352 position width for writing multiple-text-column output. If
8353 this variable contains a string representing a decimal
8354 integer, the `ls` utility shall calculate how many pathname

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

8355 text columns to write (see −C) based on the width pro-
8356 vided. If **COLUMNS** is not set or invalid, an
8357 implementation-defined number of column positions shall
8358 be assumed, based on the implementation's knowledge of
8359 the output device. The column width chosen to write the
8360 names of files in any given directory shall be constant.
8361 File names shall not be truncated to fit into the multiple-
8362 text-column output.

8363 **LANG** This variable shall determine the locale to use for the
8364 locale categories when both **LC_ALL** and the correspond-
8365 ing environment variable (beginning with **LC_**) do not
8366 specify a locale. See 2.6.

8367 **LC_ALL** This variable shall determine the locale to be used to over-
8368 ride any values for locale categories specified by the set-
8369 tings of **LANG** or any environment variables beginning
8370 with **LC_**.

8371 **LC_COLLATE** This variable shall determine the locale for character col-
8372 lation information in determining the pathname collation
8373 sequence.

8374 **LC_CTYPE** This variable shall determine the locale for the interpreta-
8375 tion of sequences of bytes of text data as characters (e.g.,
8376 single- versus multibyte characters in arguments) and
8377 which characters are defined as printable (character class
8378 `print`).

8379 **LC_MESSAGES** This variable shall determine the language in which mes-
8380 sages should be written.

8381 **LC_TIME** This variable shall determine the the format and contents
8382 for date and time strings written by `ls`.

8383 **TZ** This variable shall determine the time zone for date and
8384 time strings written by `ls`.

8385 #### 4.39.5.4 Asynchronous Events

8386 Default.

8387 ### 4.39.6 External Effects

8388 #### 4.39.6.1 Standard Output

8389 The default format shall be to list one entry per line to standard output; the
8390 exceptions are to terminals or when the −C option is specified. If the output is to a
8391 terminal, the format is implementation defined.

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

2

If the `-i` option is specified, the file's file serial number (see POSIX.1 {8} 5.6.1) shall be written in the following format before any other output for the corresponding entry: 2 2

`"%u "`, *<file serial number>* 2

If the `-l` option is specified, the following information shall be written:

`"%s %u %s %s %u %s %s\n"`, *<file mode>*, *<number of links>*, 1
<owner name>, *<group name>*, *<number of bytes in the file>*,
<date and time>, *<pathname>*

If *<owner name>* or *<group name>* cannot be determined, they shall be replaced with their associated numeric values using the format `"%u"`.

The *<date and time>*, field shall contain the appropriate date and time stamp of when the file was last modified. In the POSIX Locale, the field shall be the equivalent of the output of the following `date` command (see 4.15):

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

if the file has been modified in the last six months, or:

```
date "+%b %e  %Y"
```

(where two `<space>` characters are used between `%e` and `%Y`) if the file has not been modified in the last six months or if the modification date is in the future, except that, in both cases, the final `<newline>` produced by `date` shall not be included and the output shall be as if the `date` command were executed at the time of the last modification date of the file rather than the current time. When the LC_TIME locale category is not set to the POSIX Locale, a different format and order of presentation of this field may be used.

If the file is a character special or block special file, the size of the file may be replaced with implementation-defined information associated with the device in question.

If the pathname was specified as a *file* operand, it shall be written as specified.

The file mode written under the `-l` option shall consist of the following format:

`"%c%s%s%s%c"`, *<entry type>*, *<owner permissions>*,
<group permissions>, *<other permissions>*,
<optional alternate access method flag>

The *<optional alternate access method flag>* shall be a single `<space>` if there is no alternate or additional access control method associated with the file; otherwise, a printable character shall be used.

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

The <*entry type*> character shall describe the type of file, as follows:

d	Directory
b	Block special file
c	Character special file
p	FIFO
–	Regular file

Implementations may add other characters to this list to represent other, implementation-defined, file types.

The next three fields shall be three characters each:

<*owner permissions* > Permissions for the file owner class (see 2.9.1.3).

<*group permissions* > Permissions for the file group class.

<*other permissions* > Permissions for the file other class.

Each field shall have three character positions:

(1) If r, the file is readable; if –, it is not readable.

(2) If w, the file is writable; if –, it is not writable.

(3) The first of the following that applies:

S If in <*owner permissions*> , the file is not executable and set-user- ID mode is set. If in <*group permissions*> , the file is not executable and set-group- ID mode is set.

s If in <*owner permissions*> , the file is executable and set-user- ID mode is set. If in <*group permissions*> , the file is executable and set-group- ID mode is set.

x The file is executable or the directory is searchable.

- None of the attributes of S, s, or x applies.

Implementations may add other characters to this list for the third character position. Such additions shall, however, be written in lowercase if the file is executable or searchable, and in uppercase if it is not.

If the –l option is specified, each list of files within the directory shall be preceded by a status line indicating the number of file system blocks occupied by files in the directory in 512-byte units, rounded up to the next integral number of units, if necessary. In the POSIX Locale, the format shall be:

"total %u\n", <*number of units in the directory*>

If more than one directory, or a combination of nondirectory files and directories are written, either as a result of specifying multiple operands, or the –R option, each list of files within a directory shall be preceded by:

"\n%s:\n", <*directory name*>

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

If this string is the first thing to be written, the first <newline> character shall not be written. This output shall precede the number of units in the directory.

4.39.6.2 Standard Error

Used only for diagnostic messages.

4.39.6.3 Output Files

None.

4.39.7 Extended Description

None.

4.39.8 Exit Status

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

0 All files were written successfully.

>0 An error occurred.

4.39.9 Consequences of Errors

Default.

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

Examples, Usage

An example of a small directory tree being fully listed with ls -laRF a in the POSIX Locale:

```
total 11
drwxr-xr-x   3 hlj      prog          64 Jul  4 12:07 ./
drwxrwxrwx   4 hlj      prog        3264 Jul  4 12:09 ../
drwxr-xr-x   2 hlj      prog          48 Jul  4 12:07 b/
-rwxr--r--   1 hlj      prog         572 Jul  4 12:07 foo*

a/b:
total 4
drwxr-xr-x   2 hlj      prog          48 Jul  4 12:07 ./
drwxr-xr-x   3 hlj      prog          64 Jul  4 12:07 ../
-rw-r--r--   1 hlj      prog         700 Jul  4 12:07 bar
```

Many implementations use the equals-sign (=) and the at-sign (@) to denote sockets bound to the file system and symbolic links, respectively, for the –F option. Similarly, many historical implementations use the ‘‘s’’ character and the ‘‘l’’

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

character to denote sockets and symbolic links, respectively, as the entry type characters for the `-l` option. These characters should not be used to signify any other types of files in new implementations.

It is difficult for an application to use every part of the file modes field of `ls -l` in a portable manner. Certain file types and executable bits are not guaranteed to be exactly as shown, as implementations may have extensions. Applications can use this field to pass directly to a user printout or prompt, but actions based on its contents should generally be deferred, instead, to the `test` utility (see 4.62).

The output of `ls` (with the `-l` option) contains information that logically could be used by utilities such as `chmod` and `touch` to restore files to a known state. However, this information is presented in a format that cannot be used directly by those utilities or be easily translated into a format that can be used. In POSIX. 2, a character was added to the end of the permissions string so that applications will at least have an indication that they may be working in an area they do not understand instead of assuming that they can translate the permissions string into something that can be used. POSIX. 6 may define one or more specific characters to be used based on different standard additional or alternative access control mechanisms.

Some historical implementations of the `ls` utility show all entries in a directory except dot and dot-dot when super-user invokes `ls` without specifying the `-a` option. When ‘‘normal’’ users invoke `ls` without specifying `-a`, they should not see information about any files with names beginning with period unless they were named as file operands.

As with many of the utilities that deal with file names, the output of `ls` for multiple files or in one of the long listing formats must be used carefully on systems where file names can contain embedded white space. It is recommended that systems and system administrators institute policies and user training to limit the use of such file names.

History of Decisions Made

Implementations are expected to traverse arbitrary depths when processing the `-R` option. The only limitation on depth should be based on running out of physical storage for keeping track of untraversed directories.

The `-1` (one) option is currently found in BSD and BSD-derived implementations only. It was required in the standard so that portable applications might ensure that output is one entry per line, even if the output is to a terminal. Recent changes to the 2.10.2 allow numeric options.

Generally, the standard is mute about what happens when options are given multiple times. In the case of `-C`, `-l`, and `-1`, however, it does specify the results of these overlapping options. Since `ls` is one of the most aliased commands, it is important that the implementation do the correct thing. For example, if the alias were

```
alias ls="ls -C"
```

and the user typed ‘`ls -1`’,single text column output should result, not an error.

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

(The working group is aware that aliases are not included in the standard; this is just an example.)

The *SVID* defines a `-x` option for multi-text-column output sorted horizontally. The working group felt that `-x` provided only limited increased functionality over the `-C` option. The *SVID* also provides a `-m` option for a comma separated list of files. It was not provided because similar functionality (easier to parse for scripts) can be provided by the `echo` and `printf` utilities. Nonetheless, implementations considering adding new options to `ls` should look at historical BSD and System V versions of `ls` to avoid naming conflicts.

The BSD `ls` provides a `-A` option (like `-a`, but dot and dot-dot are not written out). The small difference from `-a` did not seem important enough to require both.

Implementations are allowed to make `-q` the default for terminals to prevent Trojan Horse attacks on terminals with special escape sequences. This is not required because:

— Some control characters may be useful on some terminals; for example, a system might write them as `\001` or `^A`,

— Special behavior for terminals is not relevant to application portability.

The `-s` option provided by existing implementations is not required by this standard. The number of disk blocks occupied by the file that it reports varies depending on underlying file system type, block size units reported, and the method of calculating the number of blocks. On some file system types, the number is the actual number of blocks occupied by the file (counting indirect blocks and ignoring holes in the file); on others it is calculated based on the file size (usually making an allowance for indirect blocks, but ignoring holes). The former is probably more useful, but depends on information not required by POSIX .1 {8}and not readily accessible on some file system types. Therefore, applications cannot depend on `-s` to provide any portable information. Implementations are urged to continue to provide this option, but applications should use the file size reported by the `-l` option in any calculations about the space needed to store a file.

An earlier draft specified that the optional alternate access method flag had to be ‘‘+’’if there was an alternate access method used on the file or `<space>` if there was not. This was changed in Draft 10 to be `<space>` if there is not and a single printable character if there is. This was done for three reasons: 1) There are existing implementations using characters other than ‘‘+’’2) There are implementations that vary this character used in that position to distinguish between various alternate access methods in use, and; 3) the developers of the standard did not want to preclude specification by POSIX. 6 that might need a way to specify more than one alternate access method. Nonetheless, implementations providing a single alternate access method are encouraged to use ‘‘+’’.

In a previous draft the units used to specify the number of blocks occupied by files in a directory in an `ls -l` listing was implementation defined. This was because BSD systems have historically used 1024-byte units and System V systems have historically used 512-byte units. It was pointed out by developers at Berkeley

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

that BSD has used 512-byte units in some places and 1024-byte units in other places. (System V has consistently used 512.) Therefore, POSIX. 2 and POSIX. 2a usually specify 512 and that value has been restored here as it was in Draft 9. Future releases of BSD are expected to consistently provide 512 as a default with a way of specifying 1024-byte units where appropriate.

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

4.40 `mailx` — Process messages

4.40.1 Synopsis

`mailx [-s` *subject* `]` *address* ...

4.40.2 Description

The `mailx` utility shall read standard input and send it to one or more addresses in an unspecified manner. Unless the first character of one or more lines is tilde (~), all characters in the input message shall appear in the delivered message, but additional characters may be inserted in the message before it is retrieved.

4.40.3 Options

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

The following option shall be supported by the implementation:

`-s` *subject* A string representing the subject of the message. All characters in the *subject* string shall appear in the delivered message. The results are unspecified if *subject* is longer than {LINE_MAX} − 10 bytes or contains a `<newline>`.

4.40.4 Operands

The following operand shall be supported by the implementation:

address Send a message to *address* . Valid login names on the local system shall be accepted as valid *address* es. The interpretation of other types of *address* es is unspecified. An implementation-defined way for a user with a login-name address to retrieve the message shall be provided by the implementation.

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

8612 ### 4.40.5 External Influences

8613 #### 4.40.5.1 Standard Input

8614 The standard input shall be a text file. The results are unspecified if the first
8615 character of any input line is a tilde (~).

8616 #### 4.40.5.2 Input Files

8617 None.

8618 #### 4.40.5.3 Environment Variables

8619 The following environment variables shall affect the execution of `mailx`:

8620 **DEAD** This variable shall affect the processing of signals by
8621 `mailx`: if the application sets this variable to `/dev/null`,
8622 the results of receiving a signal are as described by this
8623 standard; they are otherwise unspecified.

8624 **HOME** This variable shall be interpreted as a pathname of the
8625 user's home directory.

8626 **LANG** This variable shall determine the locale to use for the
8627 locale categories when both **LC_ALL** and the correspond-
8628 ing environment variable (beginning with **LC_**) do not
8629 specify a locale. See 2.6.

8630 **LC_ALL** This variable shall determine the locale to be used to over-
8631 ride any values for locale categories specified by the set-
8632 tings of **LANG** or any environment variables beginning
8633 with **LC_**.

8634 **LC_CTYPE** This variable shall determine the locale for the interpreta-
8635 tion of sequences of bytes of text data as characters (e.g.,
8636 single- versus multibyte characters in arguments and
8637 input files).

8638 **LC_MESSAGES** This variable shall determine the language in which mes-
8639 sages should be written.

8640 **MAILRC** This variable shall affect the startup processing of `mailx`:
8641 if the application sets this variable to `/dev/null`, `mailx`
8642 shall operate as described by this standard; otherwise,
8643 unspecified results occur.

8644 #### 4.40.5.4 Asynchronous Events

8645 Default.

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

8646 ### 4.40.6 External Effects

8647 #### 4.40.6.1 Standard Output

8648 None.

8649 #### 4.40.6.2 Standard Error

8650 Used only for diagnostic messages.

8651 #### 4.40.6.3 Output Files

8652 None.

8653 ### 4.40.7 Extended Description

8654 None.

8655 ### 4.40.8 Exit Status

8656 The `mailx` utility shall exit with one of the following values:

8657 0 Successful completion.

8658 >0 An error occurred.

8659 ### 4.40.9 Consequences of Errors

8660 Default.

8661 ### 4.40.10 Rationale. *(This subclause is not a part of P1003.2)*

8662 **Usage, Examples**

8663 The intent is that a header indicating who sent the message and a message sub-
8664 ject string, the contents of the standard input, and perhaps a trailer is delivered
8665 to users specified by the given addresses. The standard input, however, may have
8666 to be manipulated slightly to avoid confusion between message text and headers
8667 as it passes through the message delivery system. POSIX. 2 does not specify how
8668 standard input may be manipulated; that will be specified in detail by POSIX. 2a.

8669 The restriction on a subject line being {LINE _MAX} – 10 bytes is based on the his- 2
8670 torical format that consumes 10 bytes for `"Subject: "` and the trailing `<new-` 2
8671 `line>`. Many historical mailers that a message may encounter on other systems 2
8672 will not be able to handle lines that long, however. 2

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

8673 **History of Decisions Made**

8674 The developers of the standard felt strongly that a method for applications to send
8675 messages to specific users was necessary. The obvious example is a batch utility,
8676 running noninteractively, that wishes to communicate errors or results to a user.
8677 However, the actual format, delivery mechanism, and method of reading the mes-
8678 sage are clearly beyond the scope of this standard.

8679 The intent of this command is to provide a simple, portable interface for sending
8680 messages noninteractively. It merely defines a ''front-end'' to the historical mail
8681 system. It is suggested that implementations explicitly denote the sender and
8682 recipient in the body of the delivered message. Further specification of formats
8683 for either the message envelope or the message itself were deliberately not made,
8684 as the industry is in the midst of changing from the current standards to a more
8685 internationalized standard and it is probably incorrect, at this time, to require
8686 either one.

8687 Implementations are encouraged to conform to the various delivery mechanisms
8688 described in ARPANET Requests for Comment Numbers 819, 822, 882, 920, 921,
8689 and the CCITT X.400 standards.

8690 The standard does not place any restrictions on the length of messages handled
8691 by `mailx`, and for delivery of local messages the only limitations should be the
8692 normal problems of available disk space for the target mail file. When sending
8693 messages to external machines, applications are advised to limit messages to less
8694 than 50 kilobytes because many mail gateways impose message-length restric-
8695 tions. (Note that this is usually an administrative issue based on the amount of
8696 mail traffic and disk space available on the gateways. Therefore, there is no way
8697 for this standard to require implementations to guarantee delivery of long mes-
8698 sages to remote systems.)

8699 Like the utilities `logger` and `lp`, `mailx` is admittedly difficult to test. This was
8700 not deemed sufficient justification to exclude these utilities from the standard. It
8701 is also arguable that they are, in fact, testable, but that the tests themselves are
8702 not portable.

8703 Before Draft 7, there was a utility named `mailto`. In Draft 7, the name was
8704 changed to `sendto` because of comments noting that `mailto` implied full mail-
8705 like functionality and that was not what the specification provided. However,
8706 there have been consistent comments that it does not make sense to end up with a
8707 standard that will require two mail-sending interfaces. (POSIX.2a is working on a
8708 fully fleshed-out mail-sending and -reading utility based on the historical
8709 System V `mailx` utility.) A message- (or mail-) sending utility that is a subset of
8710 the interactive utility that will be described by POSIX.2a is much more consistent
8711 with the rest of the standard. Therefore, in Draft 10 the name has been changed
8712 again to `mailx` and the description is a small subset of the functionality being
8713 specified by POSIX.2a. It provides a portable way for a shell script to be able to
8714 send a message to a user on the local system. It is expected that implementations
8715 that have provided `mailx` in the past will use it to meet the POSIX.2 require-
8716 ments. Implementations that have not provided `mailx` in the past will be able to
8717 create a simple interface to their current mailer to meet these requirements.

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

Most of the features provided by `mailx` (and the similar BSD `Mail`) utility are not specified here because they are not needed for noninteractive use (applications do not usually read mail without user participation) and they depend on other interactive features that are not defined by POSIX.2, but will be defined by POSIX.2a (the `~v` command, for instance, uses the `vi` editor as a default.)

If the **DEAD** environment variable is not set to `/dev/null`, historical versions of `mailx` and `Mail` save a message being constructed in a file under some circumstances when some asynchronous events occur. The details will be specified by POSIX.2a.

If the **MAILRC** environment variable does not name an empty file, historical versions of `mailx` and `Mail` read initialization commands from a file before processing begins. Since the initialization that a user specifies could alter the contents of messages an application is trying to send, applications are advised to set **MAILRC** to `/dev/null`. POSIX.2a will specify details on the format of the initialization file.

Options to specify addresses as ‘‘cc’’(carbon-copy) or ‘‘bcc’’(blind-carbon-copy) were considered to be format details and were omitted.

A zero exit status implies that all messages were *sent*, but it gives no assurances that any of them were actually *delivered*. The reliability of the delivery mechanism is unspecified and is an appropriate marketing distinction between systems.

4.41 `mkdir` — Make directories

4.41.1 Synopsis

`mkdir [-p][-m` *mode* `]` *dir* ...

4.41.2 Description

The `mkdir` utility shall create the directories specified by the operands, in the order specified.

For each *dir* operand, the `mkdir` utility shall perform actions equivalent to the POSIX.1 {8}*mkdir*() function, called with the following arguments:

(1) The *dir* operand is used as the *path* argument.

(2) The value of the bitwise inclusive OR of S_IRWXU, S_IRWXG, and S_IRWXO is used as the *mode* argument. (If the `-m` option is specified, the *mode* option-argument overrides this default.)

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

8750 ### 4.41.3 Options

8751 The `mkdir` utility shall conform to the utility argument syntax guidelines
8752 described in 2.10.2.

8753 The following options shall be supported by the implementation:

8754 `-m` *mode* Set the file permission bits of the newly-created directory to the
8755 specified *mode* value. The *mode* option-argument shall be the
8756 same as the *mode* operand defined for the `chmod` utility (see 4.7).
8757 In the *symbolic_mode* strings, the *op* characters `+` and `-` shall be
8758 interpreted relative to an assumed initial mode of `a=rwx`; `+` shall
8759 add permissions to the default mode, `-` shall delete permissions
8760 from the default mode.

8761 `-p` Create any missing intermediate pathname components.

8762 For each *dir* operand that does not name an existing directory,
8763 effects equivalent to those caused by following command shall
8764 occur:

8765 `mkdir -p -m $(umask -S),u+wx $(dirname` *dir* `) &&`
8766 `mkdir [-m` *mode* `]` *dir*

8767 where the `[-m` *mode* `]` option represents that option supplied to
8768 the original invocation of `mkdir`, if any.

8769 Each *dir* operand that names an existing directory shall be
8770 ignored without error.

8771 ### 4.41.4 Operands

8772 The following operand shall be supported by the implementation:

8773 *dir* A pathname of a directory to be created.

8774 ### 4.41.5 External Influences

8775 #### 4.41.5.1 Standard Input

8776 None.

8777 #### 4.41.5.2 Input Files

8778 None.

8779 #### 4.41.5.3 Environment Variables

8780 The following environment variables shall affect the execution of `mkdir`:

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

8781 **LANG** This variable shall determine the locale to use for the
8782 locale categories when both **LC_ALL** and the correspond-
8783 ing environment variable (beginning with **LC_**) do not
8784 specify a locale. See 2.6.

8785 **LC_ALL** This variable shall determine the locale to be used to over-
8786 ride any values for locale categories specified by the set-
8787 tings of **LANG** or any environment variables beginning
8788 with **LC_**.

8789 **LC_CTYPE** This variable shall determine the locale for the interpreta-
8790 tion of sequences of bytes of text data as characters (e.g.,
8791 single- versus multibyte characters in arguments).

8792 **LC_MESSAGES** This variable shall determine the language in which mes-
8793 sages should be written.

8794 #### 4.41.5.4 Asynchronous Events

8795 Default.

8796 ### 4.41.6 External Effects

8797 #### 4.41.6.1 Standard Output

8798 None.

8799 #### 4.41.6.2 Standard Error

8800 Used only for diagnostic messages.

8801 #### 4.41.6.3 Output Files

8802 None.

8803 ### 4.41.7 Extended Description

8804 None.

8805 ### 4.41.8 Exit Status

8806 The `mkdir` utility shall exit with one of the following values:

8807 0 All the specified directories were created successfully or the `-p` option
8808 was specified and all the specified directories now exist.

8809 >0 An error occurred.

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

8810 ### 4.41.9 Consequences of Errors

8811 Default.

8812 ### 4.41.10 Rationale. *(This subclause is not a part of P1003.2)*

8813 **Examples, Usage**

8814 The default file mode for directories is `a=rwx` (777) with selected permissions
8815 removed in accordance with the file mode creation mask. For intermediate path
8816 name components created by `mkdir`, the mode is the default modified by `u+wx` so
8817 that the subdirectories can always be created regardless of the file mode creation
8818 mask; if different ultimate permissions are desired for the intermediate direc-
8819 tories, they can be changed afterward with `chmod`.

8820 Application writers should note that some of the requested directories may have
8821 been created even if an error occurs.

8822 **History of Decisions Made**

8823 The System V `-m` option was added to control the file mode.

8824 The System V `-p` option was added to create any needed intermediate directories,
8825 to complement the functionality provided `rmdir` for removing directories in the
8826 path prefix as they become empty. Because no error is produced if any path com-
8827 ponent already exists, the `-p` option is also useful to ensure that a particular
8828 directory exists.

8829 The functionality of `mkdir` is described substantially through a reference to the
8830 *mkdir* () function in POSIX.1 {8}. For example, by default, the mode of the direc-
8831 tory is affected by the file mode creation mask in accordance with the specified
8832 behavior of POSIX.1 {8} *mkdir* (). In this way, there is less duplication of effort
8833 required for describing details of the directory creation.

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

4.42 mkfifo — Make FIFO special files

4.42.1 Synopsis

mkfifo [−m *mode*] *file* ...

4.42.2 Description

The mkfifo utility shall create the FIFO special files specified by the operands, in the order specified.

For each *file* operand, the mkfifo utility shall perform actions equivalent to the POSIX .1 {8}*mkfifo* () function, called with the following arguments:

(1) The *file* operand is used as the *path* argument.

(2) The value of the bitwise inclusive OR of S_IRUSR , S_IWUSR , S_IRGRP , S_IWGRP , S_IROTH , and S_IWOTH is used as the *mode* argument. (If the −m option is specified, the *mode* option-argument overrides this default.)

4.42.3 Options

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

The following option shall be supported by the implementation:

−m *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument shall be the same as the *mode* operand defined for the chmod utility (see 4.7). In the *symbolic_mode* strings, the *op* characters + and - shall be interpreted relative to an assumed initial mode of a=rw.

4.42.4 Operands

The following operand shall be supported by the implementation:

file A pathname of the FIFO special file to be created.

4.42.5 External Influences

4.42.5.1 Standard Input

None.

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

8861 #### 4.42.5.2 Input Files

8862 None.

8863 #### 4.42.5.3 Environment Variables

8864 The following environment variables shall affect the execution of `mkfifo`:

8865 **LANG** This variable shall determine the locale to use for the
8866 locale categories when both **LC_ALL** and the correspond-
8867 ing environment variable (beginning with **LC_**) do not
8868 specify a locale. See 2.6.

8869 **LC_ALL** This variable shall determine the locale to be used to over-
8870 ride any values for locale categories specified by the set-
8871 tings of **LANG** or any environment variables beginning
8872 with **LC_**.

8873 **LC_CTYPE** This variable shall determine the locale for the interpreta-
8874 tion of sequences of bytes of text data as characters (e.g.,
8875 single- versus multibyte characters in arguments).

8876 **LC_MESSAGES** This variable shall determine the language in which mes-
8877 sages should be written.

8878 #### 4.42.5.4 Asynchronous Events

8879 Default.

8880 ### 4.42.6 External Effects

8881 #### 4.42.6.1 Standard Output

8882 None.

8883 #### 4.42.6.2 Standard Error

8884 Used only for diagnostic messages.

8885 #### 4.42.6.3 Output Files

8886 None.

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

4.42.7 Extended Description

None.

4.42.8 Exit Status

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

0 All the specified FIFO special files were created successfully.

>0 An error occurred.

4.42.9 Consequences of Errors

Default.

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

Examples, Usage

None.

History of Decisions Made

This new utility was added to permit shell applications to create FIFO special files.

The `-m` option was added to control the file mode, for consistency with the similar functionality provided the `mkdir` utility.

Earlier drafts included a `-p` option similar to `mkdir`'s `-p` option that created intermediate directories leading up to the FIFO specified by the final component. This was removed because it is not commonly needed and is not common practice with similar utilities.

The functionality of `mkfifo` is described substantially through a reference to the *mkfifo* () function in POSIX. 1. For example, by default, the mode of the FIFO file is affected by the file mode creation mask in accordance with the specified behavior of POSIX .1 {8} *mkfifo* (). In this way, there is less duplication of effort required for describing details of the file creation.

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