IEEE P1003.2 Draft 11.2 – September 1991

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, 1
the *mode* option-argument overrides this default.) 1

4.41.3 Options

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

The following options shall be supported by the implementation:

`−m` *mode* Set the file permission bits of the newly-created directory 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=rwx`; `+` shall add permissions to the default mode, `-` shall delete permissions from the default mode.

`−p` Create any missing intermediate pathname components.

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

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

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

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

4.41.4 Operands

The following operand shall be supported by the implementation:

dir A pathname of a directory to be created.

4.41.5 External Influences

4.41.5.1 Standard Input

None.

4.41.5.2 Input Files

None.

4.41.5.3 Environment Variables

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

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

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

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

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

4.41.5.4 Asynchronous Events

Default.

4.41.6 External Effects

4.41.6.1 Standard Output

None.

4.41.6.2 Standard Error

Used only for diagnostic messages.

4.41.6.3 Output Files

None.

4.41.7 Extended Description

None.

4.41.8 Exit Status

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

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

>0 An error occurred.

4.41.9 Consequences of Errors

Default.

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

Examples, Usage

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

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

History of Decisions Made

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

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

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

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.