--
--

IEEE P1003.2 Draft 11.2 − September 1991

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 2
use of `−d` with `−R` produces unspecified results. 2

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

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

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

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

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

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

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

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

4.39.4 Operands

The following operands shall be supported by the implementation:

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

4.39.5 External Influences

4.39.5.1 Standard Input

None.

4.39.5.2 Input Files

None.

4.39.5.3 Environment Variables

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

COLUMNS This variable shall determine the user's preferred column position width for writing multiple-text-column output. If this variable contains a string representing a decimal integer, the `ls` utility shall calculate how many pathname text columns to write (see `-C`) based on the width provided. If **COLUMNS** is not set or invalid, an implementation-defined number of column positions shall be assumed, based on the implementation's knowledge of the output device. The column width chosen to write the names of files in any given directory shall be constant. File names shall not be truncated to fit into the multiple-text-column output.

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_COLLATE This variable shall determine the locale for character collation information in determining the pathname collation sequence.

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) and which characters are defined as printable (character class `print`).

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

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

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

4.39.5.4 Asynchronous Events

Default.

4.39.6 External Effects

4.39.6.1 Standard Output

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

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:

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

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>*,
<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.

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>*

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`' 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.

(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 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 2
in the *subject* string shall appear in the delivered message. The 2
results are unspecified if *subject* is longer than {LINE_MAX} − 10 2
bytes or contains a `<newline>`. 2

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.