IEEE P1003.2 Draft 11.2 – September 1991

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

Examples, Usage

The `logname` utility explicitly ignores the **LOGNAME** environment variable because environment changes could produce erroneous results.

History of Decisions Made

The `passwd` file is not listed as required, because the implementation may have other means of mapping login names.

4.38 `lp` — Send files to a printer

4.38.1 Synopsis

`lp` [`−c`] [`−d` *dest*] [`−n` *copies*] [*file* . . .]

4.38.2 Description

The `lp` utility shall copy the input files to an output device in an unspecified manner. The default output destination should be to a hardcopy device, such as a printer or microfilm recorder, that produces nonvolatile, human-readable documents. If such a device is not available to the application, or if the system provides no such device, the `lp` utility shall exit with a nonzero exit status.

The actual writing to the output device may occur some time after the `lp` utility successfully exits. During the portion of the writing that corresponds to each input file, the implementation shall guarantee exclusive access to the device.

4.38.3 Options

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

The following options shall be supported by the implementation:

`−c` Exit only after further access to any of the input files is no longer required. The application can then safely delete or modify the files without affecting the output operation.

`−d` *dest* Specify a string that names the output device or destination. If `−d` is not specified, and neither the **LPDEST** nor **PRINTER** environment variable is set, an unspecified output device is used. The `−d` *dest* option shall take precedence over **LPDEST** , which in turn shall take precedence over **PRINTER** . Results are undefined when *dest* contains a value that is not a valid device or destination name.

`−n` *copies* Write *copies* number of copies of the files, where *copies* is a positive decimal integer. The methods for producing multiple copies and for arranging the multiple copies when multiple *file* operands are used are unspecified, except that each file shall be output as an integral whole, not interleaved with portions of other files.

4.38.4 Operands

The following operands shall be supported by the implementation:

file A pathname of a file to be output. If no *file* operands are specified, or if a *file* operand is −, the standard input shall be used. If a *file* operand is used, but the `−c` option is not specified, the process performing the writing to the output device may have user and group permissions that differ from that of the process invoking `lp`.

4.38.5 External Influences

4.38.5.1 Standard Input

The standard input shall be used only if no *file* operands are specified, or if a *file* operand is −. See Input Files.

4.38.5.2 Input Files

The input files shall be text files.

4.38.5.3 Environment Variables

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

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 and input files).

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

LPDEST This variable shall be interpreted as a string that names the output device or destination. If the **LPDEST** environment variable is not set, the **PRINTER** environment variable shall be used. The −d *dest* option shall take precedence over **LPDEST**. Results are undefined when −d is not specified and **LPDEST** contains a value that is not a valid device or destination name.

PRINTER This variable shall be interpreted as a string that names the output device or destination. If the **LPDEST** and **PRINTER** environment variables are not set, an unspecified output device is used. The −d *dest* option and the **LPDEST** environment variable shall take precedence over **PRINTER**. Results are undefined when −d is not specified, **LPDEST** is unset, and **PRINTER** contains a value that is not a valid device or destination name.

4.38.5.4 Asynchronous Events

Default.

4.38.6 External Effects

4.38.6.1 Standard Output

A message concerning the identification or status of the print request may be 2
written, in an unspecified format. 2

4.38.6.2 Standard Error

Used only for diagnostic messages.

4.38.6.3 Output Files

None.

4.38.7 Extended Description

None.

4.38.8 Exit Status

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

0 All input files were processed successfully.

>0 No output device was available, or an error occurred.

4.38.9 Consequences of Errors

Default.

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

Examples, Usage

Since the default destination, device type, queueing mechanisms, and acceptable forms of input are all unspecified, usage guidelines for what a portable application can do are as follows:

(1) Use the command in a pipeline, or with `-c`, so that there are no permission problems and the files can be safely deleted or modified.

(2) Limit output to text files of reasonable line lengths and printable characters and include no device-specific formatting information, such as a page description language. The meaning of ‘‘reasonable’’ in this context can only be answered as a quality of implementation issue, but should be apparent from historical usage patterns in the industry and the locale. The `pr` and `fold` utilities can be used to achieve reasonable formatting for the implementation's default page size.

Alternatively, the application can arrange its installation in such a way that requires the system administrator or operator to provide the appropriate information on `lp` options and environment variable values.

At a minimum, having this utility in the standard tells the industry that portable applications require a means to print output and provides at least a command name and **LPDEST** routing mechanism that can be used for discussions between vendors, application writers, and users. The use of ‘‘should’’ in the Description clearly shows the working group's intent, even if it cannot mandate that all systems (such as laptops) have printers.

Examples:

To print file *file*:

```
lp -c file
```

To print multiple files with headers:

```
pr file1 file2 | lp
```

On most existing implementations of `lp`, an option is provided to pass printer specific options to the daemon handling the printer. It is not specified here because the printer-specific options are widespread and in conflict, the `lp` specified here is not required to even have a queueing mechanism, and the choice of options varies widely from printer to printer. Nonetheless, implementors are encouraged to use this mechanism where appropriate:

`−o` *option* Specifies an implementation-defined option that controls the specific operation of the printer. The following *option* s could be used for the meanings below if the hardware is capable of supporting the option.

option	Meaning
`lp2`	two logical pages per physical page
`lp4`	four logical pages per physical page
`d`	double sided

POSIX. 2 does not specify what the ownership of the process performing the writ- 1
ing to the output device may be. If `−c` is not used, it is unspecified whether the 1
process performing the writing to the output device will have permission to read 1
file if there are any restrictions in place on who may read *file* until after it is 1
printed. Also, if `−c` is not used, the results of deleting *file* before it is printed are 1
unspecified. 1

History of Decisions Made

The `lp` utility was designed to be a basic version of a utility that is already available in many historical implementations. The working group felt that it should be implementable simply as:

```
cat "$@" > /dev/lp
```

after appropriate processing of options, if that is how the implementation chose to do it and if exclusive access could be granted (so that two users did not write to the device simultaneously). Although in the future the working group may add other options to this utility, it should always be able to execute with no options or operands and send the standard input to an unspecified output device.

The standard makes no representations concerning the format of the printed output, except that it must be ‘‘human-readable’’ and ‘‘nonvolatile.’’ Thus, writing by default to a disk or tape drive or a display terminal would not qualify. (Such destinations are not prohibited when `−d` *dest* , **LPDEST** , or **PRINTER** are used, however.)

A portable application will use one of the *file* operands only with the `−c` option or if the file is publicly readable and guaranteed to be available at the time of printing. This is because the standard gives the implementation the freedom to queue up the request for printing at some later time by a different process that might not be able to access the file.

The standard is worded such that a ‘‘print job’’ consisting of multiple input files, possibly in multiple copies, is guaranteed to print so that any one file is not jumbled up with another, but there is no statement that all the files or copies have to 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.

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