-- --

IEEE P1003.2b Draft 2 – December 1991

⇒ **4.39.6.1 `ls` Standard Output.** *Add the following to the list of <entry type> characters:*

`l` (ell) Symbolic link

4.43 `mv` — Move files

⇒ **4.43.2 `mv` Description.** *Replace the first two paragraphs of the Description with:*

In the first synopsis form, the `mv` utility shall move the file named by the *source_file* operand to the *destination* specified by the *target_file*. This first synopsis form is assumed when the final operand does not name an existing directory and is not a symbolic link referring to an existing directory.

In the second synopsis form, `mv` shall move each file named by a *source_file* operand to a *destination* file in the existing directory named by the *target_dir* operand, or referenced if *target_dir* is a symbolic link. The *destination* path for each *source_file* shall be the concatenation of the target directory, a single slash character, and the last pathname component of the *source_file*. This second form is assumed when the final operand names an existing directory.

⇒ **4.43.2 `mv` Description.** *Replace the first sentence of item (5) with:*

The file hierarchy rooted in *source_file* shall be duplicated as a file hierarchy rooted in the destination path. If *source_file* or any of the files below it in the hierarchy are symbolic links, the links themselves shall be moved, rather than any files they refer to.

Editor's Note: The following rationale will be added to E.4.43, but is kept here with `mv` *for this draft:*

`mv` Rationale. *(This subclause is not a part of P1003.2b)*

When `mv` is dealing with a single file system and *source_file* is a symbolic link, the link itself is moved as a consequence of the dependence on the POSIX.1 {8} *rename*() functionality, per the Description. Across file systems, this has to be made explicit.

4.48 pax — Portable archive interchange

⇒ **4.48.3 pax Options.** *Add the following option in the proper sorted order:*

`–L` Follow all symbolic links by traversing the the pathnames that the links reference. In following symbolic links, `pax` shall keep track of the directories visited so that it can detect infinite loops. (For example, a loop would occur if a symbolic link refers to a directory that is an ancestor.) When it detects a loop, `pax` shall write a diagnostic message to standard error and shall either recover its position in the hierarchy in an implementation-defined manner or terminate. The default behavior shall be to archive the symbolic link itself.

⇒ **4.48.3 pax Options.** *Modify the description of the* `–o` *options option as follows:*

`–o` *options*

Provide information to the implementation to modify the algorithm for extracting or writing files that is specific to the file format specified by `–x`. The value of *options* shall consist of one or more comma-separated keywords of the form:

keyword `[=`*value* `][,`*keyword* `[=`*value* `], ...]`

Within the optional *value* field, the application shall precede any literal comma with a backslash, which shall be ignored, but preserves the comma as part of *value*. Multiple `–o` options can be specified. The following keyword values of *options* shall be supported for the file formats as indicated:

`charmap=`*name*

HLJ : I have included a modification of the POSIX. 2 Draft 11 `–e` *charmap option to get the working group started on the pathname translation problem. I am not personally recommending this method, but WG 15 has indicated that something similar would be necessary.*

The *name* value shall represent either a system-provided charmap description (see 2.4.1) or the pathname of an application-provided charmap file.

With the `–w` option, filenames shall be written using only characters in the portable character set (except NUL ; see Table 2-3). Any character in the filename that is not in the portable character set shall be replaced by `pax` using the following steps; the description refers to the escape character defined in the named *charmap* (see the description of `escape_char` in 2.4.1):