--   --

IEEE P1003.2b Draft 2 − December 1991

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]

STANDARDS PROJECT

Draft Standard for Information Technology — Portable Operating System Interface (POSIX) Part 2: Shell and Utilities — Amendment 2

Sponsor

Technical Committee on Operating Systems and Application Environments
of the
IEEE Computer Society

Work Item Number: JTC 1.22.21.2

Abstract: P1003.2b is part of the POSIX series of standards for applications and user interfaces to open systems. It consists of modifications and clarifications to ISO/IEC 9945-2: 199x (IEEE Std 1003.2-199x and IEEE Std 1003.2a-199x), including support for symbolic links and a new archive/interchange format.

Keywords: API, application portability, data processing, open systems, operating system, portable application, POSIX, shell and utilities, user portability

P1003.2b / D3
February 1992

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]

February 1992 *SH XXXXX*

Editor's Notes

This section will not appear in the final document. It is used for editorial comments concerning this draft.

Draft 1 was mostly the new archive/interchange format, based on Donn Terry's work in P1003.1b. See the Introduction for other input the working group needs
to consider. 2

This first draft contained diff marks (1) to differentiate material from Donn's 2
archive proposal; see 4.48. Diff marks ‘‘2’’ denote changes from Draft 1 to Draft 2. 3
Diff marks ‘‘3’’ denote changes from Draft 2 to Draft 1. Editorial changes such as 3
typos, grammatical errors or changes, changes in cross references, and removal of editorial notes are not diff-marked. Please note that it is not always feasible to get the diff marks exactly right; they will sometimes start or end a line too soon.

This draft makes assumptions about the organization and content of POSIX.2
Draft 11.2, POSIX.2a Draft 8, and POSIX.1a Draft 5. If you are not in one of these 3
balloting groups, you can purchase either draft by contacting:

IEEE Publications
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331

1 (800) 678-IEEE
+1 (908) 562-3800 (outside US)

Since portions of this standard are meant to be modifications of the base POSIX.2 standard, the draft headings in Sections 1 through 6 have been set up to match the affected clauses and still go into the table of contents. There are therefore gaps in the clause numbers.

For various editorial reasons, there is a page number gap where the Annexes will eventually go.

Please report typographical errors to:

Hal Jespersen
POSIX Software Group
447 Lakeview Way
Redwood City, CA 94062
+1 (415) 364-3410
FAX: +1 (415) 364-4498
Email: `hlj@Posix.COM`

(Electronic mail is preferred.)

This draft is available in various electronic forms to assist the review process. Our thanks to Andrew Hume of AT&T Bell Laboratories for providing online access facilities. Note that this is a limited experiment in providing online access; future ballots may provide other forms, such as diskettes or a bulletin board arrangement, but the instructions shown here are the only methods currently available. Please also observe the additional copyright restrictions that are described in the online files.

Assuming you have access to the Internet, the scenario is approximately

```
ftp research.att.com # research's IP address is 192.20.225.2
<login as netlib; password is your email address>
cd posix/p1003.2b/d3
get toc index
binary
get p11-20.Z
```

The draft is available in several forms. The table of contents can be found in `toc`, pages containing a particular section are stored under the section number, sets of pages are stored in files with names of the form p$n-m$, and the entire draft is stored in `all`. By default, files are ASCII. A `.ps` suffix indicates PostScript. A `.Z` suffix indicates a `compress`'ed file. The file `index` contains a general description of the files available.

These files are also available via electronic mail by sending a message like

```
echo send 4.48 from posix/p1003.2b/d3 |
        mail netlib@research.att.com
```

If you use email, you should *not* ask for the compressed version. For a more complete introduction to this form of *netlib* , send the message

```
send help
```

POSIX. 2b Change History

This section is provided to track major changes between drafts.

Draft 3	[February 1992] Miscellaneous minor changes to the `pax` format, provided by Mark Brown. Symbolic link material added, based on initial proposals from Dawn Burnett, as modified by the working group. 3
Draft 2	[December 1991] Miscellaneous minor changes to the `pax` format, provided by Mark Brown. Limited online access provided as part of an IEEE Computer Society experiment. 2
Draft 1	[September 1991] Conversion of `pax` formatting from P1003.1b Draft 5 and `cpio` and `pax` from IEEE Std 1003.1-1990.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the IEEE , the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contents

TABLES

Introduction

(This Introduction is not a normative part of P1003.2b Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 2, but is included for information only.)

HLJ : To be provided.

Briefly, POSIX. 2b consists of minor changes to accomplish the following (from PAR):

(1) *Resolve international comments on ISO / IEC DIS 9945-2 and 9945-2/ DAM 1 (P1003.2 and P1003.2a).*

(2) *Resolve issues resulting from requests for interpretation of the original IEEE 1003.2 and 1003.2a documents.*

(3) *Improve the clarity, accuracy, and precision of the language in the original 1003.2 and 1003.2a documents, correcting deficiencies found in implementing systems, test suites, or applications based on the documents.*

(4) *Resolve issues identified by IEEE working groups producing functional standards (profiles) that desire finer granularity in groupings of optional utilities and features.*

(5) *Incorporate interfaces associated with new facilities being produced by the P1003.1a project, such as symbolic links. (Note: Extended or supplementary shell and utility features based on P1003.1a interfaces will be included in P1003.2b only if schedules can be arranged so that the P1003.1a supplement is available for citation as a normative reference at the time of approval of P1003.2b.)*

(6) *Assume responsibility for definition of file interchange and archiving formats from P1003.1. This would involve movement of the current section 10 in IEEE Std 1003.1-1990 and the proposed new format from P1003.1a to the clause in P1003.2 that describes the ''pax''utility.*

Related Standards Activities

Activities to extend this standard to address additional requirements are in progress, and similar efforts can be anticipated in the future.

The following areas are under active consideration at this time, or are expected to become active in the near future: [1)]

---

1) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are also available from this office.

(1) Language-independent service descriptions of POSIX.1 {8}

(2) C, Ada, and Fortran Language bindings to (1)

(3) Verification testing methods

(4) Realtime facilities

(5) Secure/Trusted System considerations

(6) Network interface facilities

(7) System Administration

(8) Graphical User Interfaces

(9) Profiles describing application- or user-specific combinations of Open Systems standards for: supercomputing, multiprocessor, and batch extensions; transaction processing; realtime systems; and multiuser systems based on historical models

(10) An overall guide to POSIX-based or related Open Systems standards and profiles

Extensions are approved as ''amendments'' or ''revisions'' to this document, following the IEEE and ISO/IEC Procedures.

Approved amendments are published separately until the full document is reprinted and such amendments are incorporated in their proper positions.

If you have interest in participating in the TCOS working groups addressing these issues, please send your name, address, and phone number to the Secretary, IEEE Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, and ask to have this forwarded to the chairperson of the appropriate TCOS working group. If you have interest in participating in this work at the international level, contact your ISO/IEC national body.

P1003.2b was prepared by the 1003.2b working group, sponsored by the Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society. At the time this standard was approved, the membership of the 1003.2b working group was as follows:

Technical Committee on Operating Systems
and Application Environments (TCOS)

Chair: Jehan-François Pâris

TCOS Standards Subcommittee

Chair:	Jim Isaak
Vice Chairs:	Ralph Barker
	Robert Bismuth
	Hal Jespersen
	Lorraine Kevra
	Pete Meier
Treasurer:	Quin Hahn
Secretary:	Shane McCarron

1003.2b Working Group Officials

Chair:	Hal Jespersen
Vice Chair:	Donald W. Cragun
Editor:	Hal Jespersen
Secretary:	Dave Grindeland

Technical Reviewers

John Q. Public	John Q. Public	Jane Doe
Jane Doe	John Q. Public	John Q. Public

Working Group

Jane Doe	John Q. Public	John Q. Public
John Q. Public	Jane Doe	John Q. Public

The following persons provided valuable input during the balloting period:

John Q. Public	Jane Doe	John Q. Public
Jane Doe	John Q. Public	John Q. Public

The following persons were members of the 1003.2b balloting group that approved the standard for submission to the IEEE Standards Board:

John Q. Public *TBD Institutional Representative*

Jane Doe	John Q. Public	John Q. Public
John Q. Public	John Q. Public	Jane Doe

When the IEEE Standards Board approved this standard on *<date to be provided>*, it had the following membership:

(to be pasted in by IEEE)

Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 2

Section 1: Revisions to General

1.1 Scope

HLJ : To be provided. See the Introduction. Proposed text is solicited.

1.2 Normative References

⇒ **1.2 Normative References.** *Add the following entries to the Normative References clause in the correct sorted sequence, disregarding the sequence numbers shown here.*

{9} ISO 1001: 1986, *Information processing—File structure and labelling of magnetic tapes for information interchange.*

Issue: *The following three standards seem to correspond to ANSI X3.14, 22, 39, and 54. Someone with access to the actual documents should check this assumption carefully.*

{10} ISO 1863: 1976, *Information processing—9-track, 12,7 mm (0.5 in) wide magnetic tape for information interchange recorded at 32 rpmm (800 rpi).*

{11} ISO 3788: 1976, *Information processing—9-track, 12,7 mm (0.5 in) wide magnetic tape for information interchange recorded at 63 rpmm (1600 rpi).*

{12} ISO 5652: 1984, *Information processing—9-track, 12,7 mm (0.5 in) wide magnetic tape for information interchange—Format and recording, using group coding at 246 cpmm (6250 cpi).*

{13} ISO/IEC 10646: . . . ,[1] *Information technology—Universal Coded Character Set (UCS).*

1.3 Conformance

There are no revisions to this clause (yet?).

HLJ : There has been some discussion within the profile groups about breaking up the full Section 4 into smaller groups of required utilities. This is the place to do that.

1.4 Test Methods 2

To Be Provided. 2

1) To be approved and published.

Section 2: Revisions to Terminology and General Requirements

Editor's Note: This section is new for Draft 3. It is not further diff-marked. 3

Editor's Note: The following material on symbolic links is related to P1003.1a/D5; the definition is from that draft verbatim. All of the symbolic link material in this and later sections is contingent on P1003.1a being approved before P1003.2b; if it is not, all of this may be put into an informative annex. When P1003.1a is approved, a number of the P1003.2 definitions copied from POSIX.1 {8} will be updated automatically.

2.2.2 General Terms

⇒ **2.2.2 General Terms.** *Modify the contents of subclause 2.2.2, General Terms, to add the following definitions in the correct sorted order [disregarding the subclause numbers shown here].*

2.2.2.201 symbolic link: A type of file that contains a pathname.

The pathname is interpolated into a pathname being resolved, during pathname resolution, to create a new pathname when it is encountered. [P1003.1a/D5]

2.9.2 Pathname Resolution

⇒ **2.9.1.8 Pathname Resolution.** *Add a new sentence at the end of the subclause:*

When resolving the pathname of file being created, the effects of symbolic links shall be undefined unless the {POSIX2_SYMLINKS} variable is in effect for the directory in which the link would be created.

Editor's Note: The following rationale will be added to E.2.9.1.8, but is kept here for this draft:

Pathname Resolution Rationale. *(This subclause is not a part of P1003.2b)*

P1003.1a now includes symbolic links in pathname resolution and a number of concepts are automatically inherited in POSIX.2 by this inclusion. The large majority of standard utilities resolve pathnames and operate on files without special arrangements for symbolic links. Because of the global POSIX.1 {8} inheritance, this entails very few modifications to utility descriptions.

2.11.3 Operands

⇒ **2.11.4 Operands (POSIX.1: Def)** *Add a new paragraph at the end of the subclause:*

> All of the standard utilities shall be consistent in their processing of symbolic links specified directly on the command line versus those found elsewhere in the file hierarchy as a consequence of their operation.

Editor's Note: The following rationale will be added to E.2.11.4, but is kept here for this draft:

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

In most cases, the standard utilities follow symbolic links to access individual files, but when they traverse file hierarchies, they stop descending when they reach a symbolic link. (The developers of the standard use the informal term ''physical walk'' to describe this case.) There are stated exceptions to this (''logical walks''), such as `cp` and `du`. However, none of the utilities treat their command-line arguments any differently than their encountered files. For example, `find` does not follow symbolic links by default. In the following example:

```
find /foo -print
```

if `/foo` is a symbolic link, no output will result, because `find` treats its arguments consistently with files found during its hierarchy traversal. When modified to:

```
find /foo -follow -print
```

the output would include `/foo` from the command line and all of the files below it, including those found in traversing symbolic links. Not all historical practice is as consistent as the POSIX.2 standard utilities in this regard.

2.13.3 Pathname Variable Values

⇒ **2.13.3 Pathname Variable Values.** *Add a new subclause, 2.13.3, Pathname Variable Values, as follows:*

The values in Table 2-100 may be constants within an implementation or may vary from one pathname to another.

Table 2-100 – Pathname Variable Values

Name	Description
{POSIX2_SYMLINKS}	When referring to a directory. the system supports the creation of symbolic links within that directory; for nondirectory files, the meaning of {POSIX2_SYMLINKS} is undefined.

Symbolic Constants Rationale. *(This subclause is not a part of P1003.2b)*

The {POSIX2_SYMLINKS} variable indicates that the underlying operating system supports the creation of symbolic links in specific directories. Many of the POSIX.2 utilities that deal with symbolic links do not depend on this value. For example, a utility that follows symbolic links (or does not, as the case may be) will only be affected by a symbolic link if it encounters one. Presumably, a file system that does not support symbolic links will not contain any. This variable does affect such utilities as `ln –s` and `pax` that attempt to create symbolic links.

{POSIX2_SYMLINKS} was developed even though there is no comparable configuration value in P1003.1a. Since POSIX.2 does not depend on a fully conforming POSIX.1 {8} system underneath, the developers of the standard wished to allow systems in which this was an optional feature, perhaps on a file system basis.

Section 3: Revisions to Shell Command Language

There are no revisions to Section 3 (yet).

Section 4: Revisions to Execution Environment Utilities

4.5 cd — Change working directory

Editor's Note: The following rationale will be added to E.4.5, but is kept here with `cd` *for this draft:*

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

Some historical shells, such as the KornShell, took special actions when the directory name contained a dot-dot component, selecting the logical parent of the directory, rather than the actual parent directory (i.e., it moved up one level toward the / in the pathname, remembering what the user typed, rather than performing the equivalent of the C-language call

```
chdir("..");
```

In such a shell, the following commands would not be equivalent for all directories:

```
cd .. && ls                ls ..
```

This behavior is not consistent with the definition of dot-dot and is not allowed by this standard, although it would be allowed as an extension if triggered by an environment variable or command-line option.

4.6 `chgrp` — Change file group ownership

⇒ **4.6.3 `chgrp` Options.** *Add the following sentence to the end of the paragraph describing the* `−R` *option:*

> When a symbolic link is encountered in the file hierarchy, `chgrp` shall modify the file referenced by the link and shall continue with the other files in the directory containing the link, not attempting to follow the link to another portion of the file hierarchy.

Editor's Note: The following rationale will be added to E.4.6, but is kept here with `chgrp` *for this draft:*

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

When `chgrp` is given a symbolic link as a *file* operand, it changes the file resolved from the link. This is a consequence of the dependence on the *chown* () functionality, per the Description. The POSIX.1 {8} *chown* () function does not modify the link itself [and provides no *lchown* () function that does that]. However, in traversing directories for the `−R` option, it does not follow links into other parts of the hierarchy. This is based on historical practice. Some systems provide a `−h` option that does modify the link itself and this is a valid extension, but it has been omitted from POSIX.2 because POSIX.1 {8} does not support it. No known systems cause `−R` to follow symbolic links; this functionality is possible with

```
find ... -follow ... | xargs chgrp ...
```

4.7 `chmod` — Change file modes

⇒ **4.7.3 `chmod` Options.** *Add the following sentence to the end of the paragraph describing the* `−R` *option:*

When a symbolic link is encountered in the file hierarchy, `chmod` shall modify the file referenced by the link and shall continue with the other files in the directory containing the link, not attempting to follow the link to another portion of the file hierarchy.

⇒ **4.7.4 `chmod` Operands.** *Add the following sentence to the end of the paragraph describing the file operand:*

If *file* is a symbolic link, `chmod` shall change the mode of the file referenced by the symbolic link, not the symbolic link itself.

Editor's Note: The following rationale will be added to E.4.7, but is kept here with `chmod` *for this draft:*

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

When `chmod` is given a symbolic link as a *file* operand, it changes the file resolved from the link. Unlike the `chgrp` and `chown` descriptions, where this behavior is a consequence of the dependence on the *chown* () functionality, this must be explicitly stated for `chmod`. The POSIX.1 {8} *chmod* () function, on which many `chmod` implementations will be based, does not modify the link itself [and provides no *lchmod* () function that does that]. However, in traversing directories for the `−R` option, it does not follow links into other parts of the hierarchy. No known systems cause `−R` to follow symbolic links; this functionality is possible with

```
find ... -follow ... | xargs chmod ...
```

4.8 chown — Change file ownership

⇒ **4.8.3 chown Options.** *Add the following sentence to the end of the paragraph describing the* −R *option:*

> When a symbolic link is encountered in the file hierarchy, chown shall modify the file referenced by the link and shall continue with the other files in the directory containing the link, not attempting to follow the link to another portion of the file hierarchy.

Editor's Note: The following rationale will be added to E.4.8, but is kept here with chown *for this draft:*

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

When chown is given a symbolic link as a *file* operand, it changes the file resolved from the link. This is a consequence of the dependence on the *chown* () functionality, per the Description. The POSIX.1 {8} *chown* () function does not modify the link itself [and provides no *lchown* () function that does that]. However, in traversing directories for the −R option, it does not follow links into other parts of the hierarchy. This is based on historical practice. Some systems provide a −h option that does modify the link itself and this is a valid extension, but it has been omitted from POSIX.2 because POSIX.1 {8} does not support it. No known systems cause −R to follow symbolic links; this functionality is possible with

```
find ... -follow ... | xargs chown ...
```

4.13 `cp` — Copy files

⇒ **4.13.2 `cp` Description.** *Add the following sentence to the end of the first sentence of the seventh paragraph (beginning* ''In the following description, source_file refers . . . ''*):*

When *source_file* is a symbolic link, `cp` shall take actions based on the file or directory referenced by the link, not the contents of the link itself. In traversing the file hierachy with `−R` or `−r`, it also shall follow symbolic links to other parts of the hierarchy.

Editor's Note: Question for working group: Should the language about infinite loop detection be copied from `find` *and* `pax`*?*

Editor's Note: The following rationale will be added to E.4.13, but is kept here with `cp` *for this draft:*

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

When `cp` encounters a symbolic link to be copied, it copies the contents of the file resolved from the link. Unlike many of the other utilities, such as `chown` or the default case of `find`, `cp −r` and `−R` do traverse directories by following links into other parts of the hierarchy. This allows the data in a file hierarchy to be duplicated at another place, such as for backup purposes.

4.24 `find` — Find files

⇒ **4.24.4 `find` Operands.** *Add the following primary in the proper sorted order:*

`−follow` The primary always shall evaluate as true; it shall cause `find` to traverse directories by following symbolic links into other parts of the hierarchy. By default, `find` shall not follow symbolic links. In following symbolic links, `find` 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, `find` shall write a diagnostic message to standard error and shall either recover its position in the hierarchy in an implementation-defined manner or terminate.

⇒ **4.24.4 `find` Operands.** *In the* `−type` *c description, add the character* `l` *(ell) to represent a symbolic link.*

4.33 `ln` — Link files

⇒ **4.33.1 `ln` Synopsis.** *Modify the Synopsis to be:*

```
ln [−f] source_file target_file
ln [−f] source_file . . . target_dir
ln −s source_file target_file
ln −s source_file . . . target_dir
```

⇒ **4.33.2 `ln` Description.** *Modify the entire Description subclause as follows:*

The *ln* () utility creates either hard or symbolic links, depending on the presence of the `−s` option: hard links shall be created by default, symbolic links with `−s`.

4.33.2.1 Creating Hard Links

<current contents of 4.33.2 . . . >

If *source_file* is a symbolic link, `ln` shall follow the link and create a hard link at *destination* to the file referenced by *source_file* . If either *target_file* or *target_dir* is a symbolic link, the third or fourth synopsis forms (respectively) shall be assumed, even though `−s` was not specified; use of `−f` in these cases is undefined.

4.33.2.2 Creating Symbolic Links

In the third synopsis form, the `ln` utility shall create a symbolic link for the file specified by the *source_file* operand, at the *destination* path specified by the *target_file* operand. If *target_file* is a symbolic link to a nondirectory file, `ln` shall overwrite the symbolic link. This third synopsis form shall be assumed when there are two operands and the second is not an existing directory.

In the fourth synopsis form, the `ln` utility shall create a symbolic link for each file specified by a *source_file* operand, at a *destination* path in the existing directory named by *target_dir*. The corresponding destination path for each *source_file* shall be the concatenation of the target directory pathname, a slash character, and the last pathname component of the *source_file*. If *target_dir* is a symbolic link to a directory, `ln` shall follow the link and create a symbolic link to *source_file* in the directory *target_dir*.

For both the third and fourth synopsis forms:

— If the directory in which the destination is to reside does not support {POSIX 2_SYMLINKS }, results are undefined.

— The *source_file* operand can be any pathname and need not represent an existing or accessible file.

— The *source_file* and *destination* need not reside on the same file system.

— If the last operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is implementation defined.

For each *source_file*:

(1) If the *destination* path exists, `ln` shall write a diagnostic message to standard error, do nothing more with the current *source_file,* and go on to any remaining *source_file* s.

(2) Actions shall be performed equivalent to the POSIX.1 {8} *symlink* () function using *source_file* as the *pname* argument, and the *destination* path as the *slink* argument.

If *source_file* is itself a symbolic link, `ln` shall not follow the link, but shall create a symbolic link in *destination* to *source_file* (the symbolic link) itself.

⇒ **4.33.3 `ln` Options.** *Add the following option in the proper sorted order:*

`-s` Create symbolic links instead of the default hard links.

⇒ **4.33.4 `ln` Operands.** *Replace all of the Operands subclause with:*

The following operands shall be supported by the implementation:

source_file A pathname of a file to be linked. When `-s` is not specified this can be a regular or special file; whether a directory can be linked is implementation defined. When `-s` is specified, any pathname can be used.

target_file The pathname of the new directory entry or symbolic link to be created.

target_dir A pathname of an existing directory in which the new directory entries or symbolic links are to be created.

HLJ: Is it correct that `-f` *cannot be used with* `-s`*?*

4.39 `ls` — List directory contents

⇒ **4.39.1 `ls` Synopsis.** *Modify the Synopsis to be:*

`ls [−CFLRacdilqrtu1]` [*file* . . .]

⇒ **4.39.3 `ls` Options.** *Replace the description of the* `−F` *option with the following:*

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

⇒ **4.39.3 `ls` Options.** *Add the following option in the proper sorted order:*

`−L` When a symbolic link is encountered, write the name of the file or directory the link refers to, rather than the name of the link itself.

⇒ **4.39.6.1 `ls` Standard Output.** *Replace the six-line description of* `−l` *(beginning with* ''If the −l option is specified, . . . '')*with:*

If the `−l` option is specified without `−L`, 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 the file is a symbolic link, this information shall be about the link itself and the *<pathname>* field shall be of the form:

`"%s -> %s"`, *<name of link>* , *<contents of link>*

If both `−l` and `−L` are 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 of link>*

where all fields except *<pathname of link>* shall be for the file resolved from the symbolic link.

In both of the preceding `−l` forms, if *<owner name>* or *<group name>* cannot be determined, they shall be replaced with their associated numeric values using the format `"%u"`.

⇒ **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:* 3

–L Follow all symbolic links by traversing the the pathnames that the 3
links reference. In following symbolic links, pax shall keep track of 3
the directories visited so that it can detect infinite loops. (For exam- 3
ple, a loop would occur if a symbolic link refers to a directory that is 3
an ancestor.) When it detects a loop, pax shall write a diagnostic 3
message to standard error and shall either recover its position in the 3
hierarchy in an implementation-defined manner or terminate. The 3
default behavior shall be to archive the symbolic link itself. 3

⇒ **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):

(1) Any less-than, greater-than, and escape characters shall have an escape character inserted into the filename immediately before the character.

(2) Any character in the filename that is not part of the portable character set shall be replaced with a string consisting of a less-than character, the symbolic name of the character from the charmap, and a greater-than character.

(3) Any escape, less-than, or greater-than characters in the replacing symbolic names shall be modified as in step (1).

If a character for which no corresponding symbolic name exists in the charmap is found in a filename, `pax` shall write a diagnostic message to standard error; whether the file is written to the archive is unspecified.

With the `−r` option, or if neither the `−r` or `−w` option is specified, the following steps shall be used:

(1) Any sequences of a less-than character, not immediately preceded by an odd number of escapes, a string, and a greater-than character not immediately pre-
ceded by an odd number of escapes, shall be modified 2
as follows: 2

(a) The leading less-than character and trailing greater-than character shall be removed.

(b) Any sequences of an escape followed by an escape, less-than, or greater-than character shall have the leading escape character removed.

(c) The sequence shall be replaced with the encoded value of the character represented by the resulting symbolic name, from the charmap. If no encoding exists, the modifications to the sequence are implementation defined.

(2) Any sequences of an escape followed by an escape, less-than, or greater-than character shall have the leading escape character removed.

`linkdata`

(Applicable to the `−x` *pax* format.) When the `−w` option is given, but the `−r` is not, write the contents of a file to the archive even when that file is merely a hard link to a file whose contents have already been written to the archive.

`nolog`

(Applicable to the `−x` *pax* format.) Suppress logging of ignored records and fields. See 4.48.7.2. Messages specifically described as error messages shall not be suppressed by this option.

`notrans` 2
Suppress translation of character data or filenames, even 2
when the character set or charmap is known. This option 2
shall override the `charmap` option described previously. 2

`volid=`*id*

(Applicable to the `−x` *pax* format.) Specify a volume identifier for the VOL1 label. See 4.48.7.5.1.2. If fewer than six octets are given for *id*, the value shall be padded at the end with `<space>`s.

⇒ **4.48.3 `pax` Options.** *Modify the description of the* `−x` *format option as follows: HLJ : The changes to this are pointers to the Extended Description instead of POSIX. 1 and the new* `pax` *format; these are diff-marked. The rest is the same as the 199x standard.*

`−x` *format* Specify the output archive format. The `pax` utility shall recognize the following formats:

`cpio` The extended `cpio` interchange format specified
in 4.48.8. The default *blocksize* for this format for 1
character special archive files shall be 5120. Implementations shall support all *blocksize* values less than or equal to 32 256 that are multiples of 512.

`pax` The interchange format specified in 4.48.7. The 1
default *blocksize* for this format for character spe- 1
cial archive files shall be 10 240. Implementa- 1
tions shall support all *blocksize* values less than 1
or equal to 32 256 that are multiples of 512. *HLJ :* 1
That was copied from cpio. Should it be 20 000 1
instead? What about multiples? 1

`ustar` The extended `tar` interchange format specified in 1
4.48.9. The default *blocksize* for this format for 1
character special archive files shall be 10 240. Implementations shall support all *blocksize* values less than or equal to 32 256 that are multiples of 512.

Implementation-defined formats shall specify a default block size as well as any other block sizes supported for character special archive files.

Any attempt to append to an archive file in a format different from the existing archive format shall cause `pax` to exit immediately with a nonzero exit status.

⇒ **4.48.6.1 `pax` Standard Output.** *Replace the first paragraph with:*

If the `−w` option is specified and neither the `−f` nor `−r` options are specified, the standard output shall be the archive formatted according to one of the
specifications in the following subclauses or some other implementation- 1
defined format. (See `−x` *format* under 4.48.3.) 1

⇒ **4.48.6.3 `pax` Output Files.** *Replace the second paragraph with:* 1

If the `−w` option is specified, but the `−r` option is not, the output file named by 1
the `−f` option argument shall be a file formatted according to one of the 1
specifications in the following subclauses or some other, implementation- 1
defined, format. 1

⇒ **4.48.7 `pax` Extended Description.** *Replace this subclause with three subclauses, as follows. This will also cause the renumbering of 4.48.8 and 4.48.9 to be 4.48.10 and 4.48.11. HLJ : This was done for the editorial reason that we cannot go too deeply into the heading levels and if this were all in 4.48.7, we would have to start at the fourth level for each format, leaving little maneuvering room.*

4.48.7 Extended Description — pax Interchange Format 1

This format is based upon ISO 1001 {9}; a magnetic tape conforming to this standard also conforms to ISO 1001. The headers of these formats are defined (by ISO 1001) to use a subset of the invariant part of the characters and codings of ISO/IEC 646 {1}. There is an exception for certain names that would reasonably be in a different or larger character set.

4.48.7.1 Definitions

The following terms are used in this subclause (4.48.7).

4.48.7.1.1 a-character:

The 57 characters:

<space> ! " % & ' ( ) * + , - . / : ; < = > ? _ 0-9 A-Z

as defined by ISO/IEC 646 {1}.

(The ISO 1001 definition is more elaborate, but equivalent.)

4.48.7.1.2 alternate character set:

A character set specified in the label records in which certain data fields may be encoded.

4.48.7.1.3 al-characters:

The a-characters (ISO 1001 {9}) plus a-z.

4.48.7.1.4 archive:

A file or medium recorded in the format specified in this subclause (4.48.7).

4.48.7.1.5 block:

When the medium is magnetic tape or another device that records information in discrete units that have a variable length: the meaning defined in ISO 1001 {9}.

> A group of octets recorded consecutively in accordance with the relevant standard for recorded magnetic tape. [ISO 1001.] 2

Otherwise, it is a quantity of data operated on as a single unit.

4.48.7.1.6 file section:

That part of a file that is recorded on any one volume. [ISO 1001.]

4.48.7.1.7 folded to a-characters:

A string of characters when converted to contain only a-characters, as defined in 4.48.7.2.

4.48.7.1.8 label:

A record that identifies and characterizes a volume, or a file section on a volume. [ISO 1001.]

4.48.7.1.9 log:

Write to `pax` standard error. 1

2

4.48.7.1.10 magnetic tape:

Media as defined by ISO 1863 {10}, ISO 3788 {11}, and ISO 5652 {12}. [ISO 1001.]

4.48.7.1.11 record (noun):

Related data treated as a unit of information. [ISO 1001.]

4.48.7.1.12 record (verb):

When discussing the content of an archive, to transfer data from the file hierarchy to the archive.

4.48.7.1.13 restore:

Transfer data from the archive to the file hierarchy.

4.48.7.1.14 tape mark:

A control block used as a delimiter. [ISO 1001.]

NOTE: The structure of tape marks is specified by the relevant standards for recorded magnetic tape.

4.48.7.1.15 volume:

When referring to magnetic tape, this shall have the same meaning as in ISO 1001 {9}:

> A dismountable reel of magnetic tape. [ISO 1001.]

When referring to other media, it shall refer to dismountable media. It shall refer to the content of a regular file when this format is recorded in a regular file.

The following notation is used in tables describing record formats:

BP	Octet position within the label 2
L	Length of the field in number of octet positions
digit(s)	Any digit from `0` through `9`.
P	When this field in a table is marked, it indicates that a more specific requirement than that of ISO 1001 {9} is made.

4.48.7.2 General Requirements

This subclause is an extension to ISO 1001 {9}; conflicts (as opposed to further specification) with ISO 1001 shall be resolved in favor of ISO 1001.

When restoring from an archive, the process invoking `pax` might not have the appropriate privileges to create files with all the characteristics described in the archive. In such a case, the file shall be created with as many of the characteristics that it had in the archive as the privileges of the process and the system implementation permit. In the case of a process that has no additional privileges, the protection information (ownership and access permissions) shall be set in the same fashion that the POSIX.1 {8} *open*() function would when given the *mode* argument matching the file permissions supplied by the mode field supplied in the File Information File (see 4.48.7.6). A process with permissions to do so shall be capable of restoring the ownership, permissions, and other file characteristics exactly as recorded on the medium.

The `pax` utility has various options that systematically change the names of files on the archive or as restored to the file hierarchy. For simplicity, the rest of this description is written as if such name changes were not made; `pax` shall operate in such a fashion that the relationship between files specified by links and symbolic links is retained in the presence of such changes, to the extent permitted by the organization of the file hierarchy.

It is implementation defined what protections are applied to an archive based upon protection fields recorded in it. Interpretation of the protection fields need not be applied until the information has been restored.

When a device that records data in discrete units of variable length is used, a *block* may be longer than the record or records it contains. The `pax` utility shall deal with the extra information in such a way that the extra octets do not participate in the data recorded on the archive.

When a device that does not support discrete units of variable length is used, a *block* shall be exactly the specified length.

When logging of information is required by this standard, the name of the file (as it appears in the FIF) shall be included in the log in such a way that it can be associated with the other logged information for that file.

Throughout this subclause, it is stated that certain fields, records, and labels ''shall be ignored.'' Such items (if present) may be recorded with any value consistent with ISO 1001 {9}. Nothing in this standard shall be construed to require that these fields shall be ignored in the presence of options to `pax` enabling interpretation of some or all of these fields. However, such fields shall be ignored when such options are not used. The existence and content of such items may be logged. Logging of ignored information can be suppressed by the `pax –o nolog` option.

A string shall be folded to a-characters to fit in a specific field in a label subject to the following constraints:

(1) The character set is converted to ISO/IEC 646 IRV {1}; characters not in ISO/IEC 646 {1} IRV shall be converted in an implementation-defined manner.

(2) The characters `a-z` are converted to `A-Z`.

(3) The string is truncated to the field width, or padded with `<space>`s on the right, as needed to make it exactly the width of the label field.

4.48.7.3 Media

The originator and recipient of the volume used for interchange shall agree on the media upon which this format is to be recorded. If that media is agreed to be magnetic tape, a tape mark shall be recorded and expected on the medium in the locations defined below. Where an analogous concept exists for other media, it should be used.

Where a tape mark or other out-of-band end-of-file marker is not defined for the agreed-to media it shall be omitted from the media when the interchange format is recorded.

4.48.7.4 File Representation

An instance of this archive format shall consist of one or more volumes. Each volume shall contain zero or more file sections. A file shall be recorded in one or more file sections consistent with the rules of ISO 1001 {9}.

Where tape marks are available, multiple File Sections and the appropriate End of File or Section and End of Volume Label Groups shall be used. An archive written on magnetic tape shall use tape marks.

On media where both a tape mark or equivalent is not used, and where records are not delimited in hardware or where this fact is hidden by the underlying system, only a single logical volume shall be written, which may span several physical volumes. Both the labels and the file data shall be combined as a single stream, and the labels shall appear immediately adjacent to each other. If the archive must span more than one physical volume, the last octet from the archive that is recorded on the first physical volume shall be the one preceding the first octet recorded on the second physical volume.

On media where a tape mark or equivalent is not used, but where records are delimited in hardware, only a single logical volume may be written, which may span several physical volumes. Each label shall appear as a separate physical record. File data shall be treated as a single stream. If the archive must span more than one physical volume, the last octet from the archive that is recorded on the first physical volume shall be the one preceding the first octet recorded on the second physical volume.

Media recorded in conformance with this standard shall have volume header and trailer labels as described below. Such media shall represent each POSIX file in either one or two ISO 1001 {9} format files. The first such ISO 1001 file shall consist of labeling information as described for the File Information File (FIF). The second section, the File Data File (FDF), is optional—when the FIF describes a file that does contain data, it shall not be recorded.

4.48.7.5 Volume Label Records

This subclause specifies how each volume label field is defined in the context of this standard.

4.48.7.5.1 The VOL1 Header Label

BP	Field Name	L	P	Content	
1−3	Label Identifier	3		`VOL`	
4	Label Number	1		`1`	
5−10	Volume identifier	6		a-characters	
11	Volume accessibility	1		a-character	
12−24	(Reserved for future standardization)	13		`<space>`s	
25−37	Implementation Identifier	13	*	a-characters	3
38−51	Owner Identifier	14	*	a-characters	
52−79	(Reserved for future standardization)	28		`<space>`s	
80	Label Standard Version	1		`4`	

4.48.7.5.1.1 Fields Reserved for Future Standardization

These fields shall be reserved for future standardization for ISO 1001 {9} and its successors. The characters in these fields shall be `<space>`s.

4.48.7.5.1.2 Volume Identifier

This field shall specify an identification of the volume. When `pax` creates a 1
volume, the `-o volid` option can be used to set this field. If this field is not set 1
explicitly it shall contain spaces. When `pax` interprets a volume, this field shall 1
be logged. 1

4.48.7.5.1.3 Volume Accessibility

HLJ : Help from 1003.6 is requested in this area.

It is unspecified whether any mechanism is present to control access to files as recorded in this format based upon any identifying information or security information recorded with the file.

4.48.7.5.1.4 Implementation Identifier

This is a constant for the archive format, as opposed to implementation defined
for ISO 1001 {9}. It is implementation defined whether `pax` will attempt to read a 1
file or other media in ISO 1001 format that does not contain the values specified
for these fields.

4.48.7.5.1.5 Owner Identifier

This field shall contain a representation of the user name associated with the *uid* of the process creating the archive file, as found in the user database. The name shall be folded to a-characters.

4.48.7.5.2 The VOL2 Header Label

This header label is optional; if omitted, the fields described in the record shall all take their default values.

BP	Field Name	L	P	Content	
1−3	Label Identifier	3		`VOL`	
4	Label Number	1		`2`	
5−9	POSIX Identification	5	*	`POSIX`	3
10	POSIX Version	1	*	`1`	3
11−15	Standards Body	5	*	a-characters	3
16−20	Number	5	*	a-characters	3
21−25	Variant	5	*	a-characters	3
26−29	Revision	4	*	a-characters	3
30−80	Reserved for POSIX use	51	*	`<space>`s	3

NOTE: The reservation of BP 30−80 is an extension from ISO 1001 {9};in that standard, labels VOL2 3
through VOL9 are reserved for implementation use. 2

4.48.7.5.2.1 Character Set Identification

The four fields Standards Body, Number, Variant, and Revision are used to identify the alternate character set used in data files, file names, and user and group names. The following are defined to refer to known standards. Additional names may be agreed on between the originator and the recipient. If a name is not recognized, `pax` may take implementation-defined actions, but there shall always 1 be a way for the data to be transferred to the receiving system, possibly without translation.

Body	Number	Variant	Revision	Formal Standard
`<space>`s	`<space>`s	`<space>`s	`<space>`s	ISO/IEC 646 IRV {1}
`ISO`	`646`	`IRV`	`1990`	Note 1
`ISO`	`8859`	`1`	`1987`	ISO 8859-1 {5}
`ISO`	`8859`	`2`	`1987`	ISO 8859-2 {6}
`ISO/IEC`	`10646`	Note 2	`19xx`	ISO/IEC 10646 {13}

NOTES:

(1) In this form of entry, when the value for variant is `IRV`, then ISO/IEC 646 IRV {1}is specified. Other values specify National Usage variants of this character set, and shall be agreed upon by sender and recipient.

(2) This value specifies the compaction method used.

HLJ : Working group input needed for the following. See my sample `−o charmap` 1
description. 1

The format-reading utility shall provide a means to control whether data files have character set translation applied. This may be done either on a per-file basis, or selecting the option for the whole archive. A utility that performs the same translation should be provided.

4.48.7.5.3 The VOL 3 Header Label 2

This label shall be used in those instances where a volume spans more than one 2
physical medium, and the normal end-of-section labels could not be written (due 2
to device/driver limitations, for example). 2

BP	Field Name	L	P	Content	
1–3	Label Identifier	3		`VOL`	2
4	Label Number	1		`3`	2
5–7	Media Continuation Identifier	3		`EOM`	2
8–11	Media Continuation Number	4		digits	2
12	FIF /FDF Identifier	1	*	(`H` or `D`)	2
13–29	File Identifier	17	*	4.48.7.6.1.1	2
30–33	File Section Number	4		4.48.7.6.1.3	2
34–38	POSIX Identification	5	*	`POSIX`	2
39	POSIX Version	1	*	`1`	2
40–45	Implementation Identification	6	*	`<space>`s	2
46–80	(Reserved for future standardization)	7		`<space>`s	2

4.48.7.5.3.1 Media Continuation Number 2

This field shall contain the count, starting from zero (0), of the physical media 2
used to write the volume. The first continuation tape shall be numbered 0001 2
(the initial tape being 0000). 2

4.48.7.5.4 The VOL 4–9 and UVL 1–9 Header Labels 2

These labels are free to be used for any implementation-defined purpose; they 1
shall be ignored on input. It is implementation-defined as to whether they are 1
included in output. 1

4.48.7.6 File Information File

This subclause specifies how the File Information File (FIF) is recorded; both its content and the header labels for it are defined. A FIF consists of ISO 1001 {9} header labels that define the content of the FIF ; the FIF itself contains the information necessary to recreate the POSIX file (except the data).

4.48.7.6.1 First FIF Header Label (HDR 1)

This record shall be recorded for all files in the archive.

BP	Field Name	L	P	Content	
1–3	Label Identifier	3		`HDR`	
4	Label Number	1		`1`	
5–21	File Identifier	17	*	a-characters	
22–27	File Set Identifier	6		a-characters	
28–31	File Section Number	4		digits	
32–35	File Sequence Number	4		digits	
36–39	Generation Number	4		digits	
40–41	Generation Version Number	2		digits	
42–47	Creation Date	6	*	See 4.48.7.6.1.7	1
48–53	Expiration Date	6	*	See 4.48.7.6.1.8	3
54	File Accessibility	1		a-characters	
55–60	Block Count	6		`000000`	1
61–73	Implementation Identification	13	*	`<space>`s	3
74–78	POSIX Identification	5	*	`POSIX`	3
79	POSIX Version	1	*	`1`	3
80	FIF Identifier	1	*	`H`	3

4.48.7.6.1.1 File Identifier

This shall be the name of the FIF file. It shall be derived from the name of the POSIX file being recorded by prefixing the string `INFO-` to the filename (not pathname) of the file being recorded, with the result folded to a-characters.

4.48.7.6.1.2 File Set Identifier

This field shall be ignored.

4.48.7.6.1.3 File Section Number

This field shall be as specified in ISO 1001 {9}.

4.48.7.6.1.4 File Sequence Number

This field shall be ignored.

4.48.7.6.1.5 Generation Number

This field shall be ignored.

4.48.7.6.1.6 Generation Version Number

This field shall be ignored.

4.48.7.6.1.7 Creation Date

This shall be the representation of the POSIX.1 {8} *st_ctime* field of the file, converted to a Julian day, according to the time zone active at the time `pax` was run. 1
The Julian date shall appear in the six octets as follows: 1

1 If the first two digits of the year are 19, this octet shall be a `<space>`; 1
otherwise, it shall be `0`. 1

2–3 These two octets shall be the two least significant digits of the year. 1

4–6 These three octets shall be the three-digit ordinal number of the day from `001` to `366`. 1 1

4.48.7.6.1.8 Expiration Date

This shall be the same value as the creation date; see 4.48.7.6.1.7. 3

4.48.7.6.1.9 File Accessibility

HLJ : Help from 1003.6 is solicited.

The content of this field may be any permissible value; `pax` shall ignore this field. 1

4.48.7.6.1.10 Block Count

This field shall be as specified in ISO 1001 {9}.

4.48.7.6.2 Second File Header Label (HDR 2)

BP	Field Name	L	P	Content	
1–3	Label Identifier	3		`HDR`	
4	Label Number	1		`2`	
5	Record Format	1	*	`D`	
6–10	Block Length	5	*	`20000`	
11–15	Record Length	5	*	`09999`	
16–20	Standards Body	5	*	a-characters	
21–25	Number	5	*	a-characters	
26–30	Variant	5	*	a-characters	
31–34	Revision	4	*	a-characters	
35–50	(Reserved for Implementation)	16		not specified	1
51–52	Offset Length	2	*	`00`	2
53–80	(Reserved for Standard)	28		`<space>`s	1

4.48.7.6.2.1 Standards Body, Number, Variant, Revision

These fields record the same information, and are encoded in the same way, as the corresponding fields in the VOL2 Header Label. They specify the alternate character set to be used in the FIF file for those fields permitted to be in the alternate character set. If all these fields are spaces, instead of indicating ISO/IEC 646 {1}IRV, the alternate character set specified in the VOL2 label shall be used.

4.48.7.6.2.2 Record Format

Data is recorded as variable length records, blocked between 20 and 20K octets per block.

4.48.7.6.2.3 Block Length

This is the maximum length of a block, which is 20 000 octets.

4.48.7.6.2.4 Record Length

This is the maximum length of a record, which is limited to 9999 octets by the structure of variable-length records.

4.48.7.6.2.5 Offset Length

This field shall be zeroes. 2

4.48.7.6.3 Additional File Header Labels (HDR 3−9, UHL ?) 2

For a FIF, these labels are free to be used for any implementation purpose, and shall be ignored on input. It is implementation-defined as to whether they are 1
written on output. On media that supports tape marks, the last header shall be followed by a tape mark. On other media the tape mark shall be omitted.

4.48.7.6.4 FIF Contents

The FIF shall consist of several fixed record types of variable length, in the order 1
specified below. Each logical record shall be recorded in ISO/IEC 646 {1} IRV, except that certain fields may be recorded in the alternate character set. Additional optional record types are permitted in the format specified below. In the record descriptions below, the content is described excluding the record length as specified in ISO 1001 {9}.

Records present in a FIF, but which are not recognized by `pax`, shall be ignored. 1

4.48.7.6.4.1 FIF Filename Record

BP	Field Name	L	Content	
1	File Type	1	al-characters	
2−10	File Mode	9	al-characters	
11	Link Status	1	al-characters	
12−31	Modification time	20	digits (with possible decimal point)	2
32−51	Creation time	20	digits (with possible decimal point)	2
52−71	Access time	20	digits (with possible decimal point)	2
72−91	File Size	20	digits	1
92−	File Name		Alternate character set	1

File Type

This is the single character as follows:

- `-` Regular file
- `c` Character special file
- `b` Block special file

`d` Directory

`p` FIFO special file

`h` High-performance file

`l` Symbolic link

`A-Z` Reserved for implementations

other Reserved for future standardization.

The interpretation of the FIF and FDF for each of these file types shall be as specified below.

File Mode

The file protections shall consist of nine characters, presented as three groups of three characters each:

<owner permissions> Permissions for the file owner class.

<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.

`T` If in *<other permissions>*, the file is executable and an unspecified concept of shared-executable is applied to the file. It is implementation defined whether such a mode is recorded. It is implementation defined what action, if any, beyond setting the the other-execute mode, is performed when `T` is found in this position in an archive.

`x` The file is executable or the directory is searchable.

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

Implementations may add other characters to this list for the third character position. Such additions shall, however, be recorded in lowercase if the file is executable or searchable, and in uppercase if it is not. It is implementation defined whether, when such an extension is found when restoring an archive, any action beyond setting the group-execute mode according to the case of the character occurs.

DT: This is a fairly conservative approach to all this; given that this is new work, should the suid, sgid, and sticky bits be expressed separately?

Link Status

This field indicates whether the file is a link or not. An ordinary file entry is indicated for the first (or only) entry in the archive for that file. Subsequent entries for additional links to that file shall indicate whether a link with or without data is recorded. The `pax –o linkdata` controls whether data is recorded; by default, 1
this linked data is not recorded. 1

When a link, with or without data, is recorded, a record containing the *target* of the link, recorded in the alternate character set, shall immediately follow the FIF File Group record, and preceding any information specific to the file type. For full portability, the length of the target shall not exceed {_POSIX_PATH_MAX} octets.

If the target of the link was restored by `pax` during the same invocation, then a 1
link to the target shall be created and the FDF associated with the link entry shall be ignored. If the link cannot be created, an error shall be reported and the link ignored.

If a file was not restored during the same run the following actions shall be taken:

	File Type Does Not Require Data (Including Zero Length)	**File Type Requires Data**
Data present	see 4.48.7.7	restore
Data absent	restore log	ignore entry log error

A link with data shall be recorded with the File Size found at the time that specific link was recorded. If the length is found to be different from the length as recorded for a previous link, that fact shall be logged.

A link without data shall have the File Size as found at the time the specific link was recorded, but no FDF shall be recorded for that file.

The link type information is recorded as the single character as specified as follows:

`<space>` Ordinary file entry

`L` Link (without data)

`D` Link (with data)

other uppercase letters Reserved for implementations 2

other reserved for future standardization.

File Modification, Access, and Creation Times

These are the decimal representation of these times, as seconds since the Epoch. 2
If a period (.) decimal point character is present, the digits to the right of the point 2
shall represent the units of an implementation-defined subsecond timing granu- 2
larity, which shall be ignored by implementations not supporting these units. 2
The access and modification times shall be restored if the process has the appropriate privilege required to do so.

File Size

For all file types except those noted below, this is the size of the file, in octets, that is recorded on the media as the FDF; if the size of the file on the originating system should change during recording of the file, the file on the archive shall be truncated or padding of octets containing zeroes added as required to assure that the exact number of octets specified is actually recorded.

Name

The rest of the record shall consist of the Name, which shall be recorded in the alternate character set. No padding of Name shall occur, so that the length of
Name can be calculated from the record length. The `pax` utility shall translate 1
the Name in an implementation-defined manner to be the name of the file on the 1
receiving system. *HLJ : That needs work.* In no case shall a file name that is not 1
supported on the receiving system, or that cannot be subsequently accessed, be created. The length shall not exceed {_POSIX_PATH_MAX} without agreement between the originator and the recipient that this can occur.

4.48.7.6.4.2 FIF File Owner Record 1

This shall be the name of the owner of the file as found in the user database on the originating system, represented in the alternate character set. The (variable-length) record shall consist solely of the the file owner name, with no padding. A zero-length record shall be recorded if the implementation creating the format does not support ownership on this file type. If a zero length owner is encountered, it shall be treated as if the entry could not be found in the data base.

After an implementation-defined translation to the character set of the receiving
system, the File Owner name shall be used as the name of the owner of the file. If
the same name appears in the user database of the receiving system (without
truncation of either name), and if `pax` has the appropriate permission to do so, it 1
shall set the owner- ID of the file it is restoring to be the corresponding user- ID. If
the owner name does not appear in the user database of the receiving system, the
file shall be created with the ownership of the user running `pax`, and the name 1
that was not found shall be logged.

If the implementation does not support ownership on the file type being restored, the owner information may be ignored.

4.48.7.6.4.3 FIF File Group Record

This shall be treated in the same way as the file owner entry except that the group database shall be consulted, and the file group owner set where appropriate.

4.48.7.6.4.4 File-type Dependent Information

The next several records in the FIF vary depending on the File Type recorded in the first record. The following subclauses describe these additional record types and the interpretation of certain of the fields in the preceding records.

Regular Files

No File-type Dependent Information is recorded. The File Size is interpreted as the number of octets found in the FDF, except that the FDF shall be omitted if the file size is zero.

Block or Character Special Files

The file size in the first FIF record shall be zero, and no FDF shall be recorded.

BP	Field Name	L	Content
1–15	System Identifier	15	al-characters
16–	Device Specification	--	al-characters

System Identifier

When written by `pax`, this field shall contain an implementation-defined value representing the name of the implementation, the model, and submodel numbers sufficient to distinguish it from similar implementations that use a different representation of the Device Specification field. If `pax` recognizes this field, it shall restore the special file according to the information appearing in the Device Specification. If `pax` does not recognize the System Identifier, the content of the FIF, including the Device Specification and System Identifier, shall be logged and `pax` may otherwise ignore the archive entry.

Device Specification

This field (of variable length) is unspecified by this standard, except that it shall contain information sufficient to create a file identical in function to the one written by `pax`, when restored on the same implementation on which it was written.

Directories

The information in a directory FIF shall be used to restore file ownership and permissions only after `pax` has completed reading the archive contents for that directory. The meaning of the file size field is implementation defined. If the implementation has no special definition for this field, it shall be ignored. No FDF shall be recorded for this entry type.

FIFO s

No additional records shall be written. The file size shall be ignored, and no FDF shall be recorded.

Symbolic Links

A single record consisting solely of the pathname of the target of the symbolic link, in the alternate character set, shall be recorded. For full portability, the length of the name shall not exceed {_POSIX_PATH_MAX} octets. The link shall be created whether or not the target of the link exists.

The ownership and protection information recorded for a symbolic link may be ignored.

If a symbolic link entry is encountered on a system that does not support symbolic links, it is implementation defined whether, and under what conditions, the link is translated to a hard link. This translation shall be logged giving both names. If the link is not restored, that fact shall be logged.

High Performance Files

HLJ : Help is solicited from P1003.4.

High-performance files shall be restored as regular files if the implementation does not support the file type. In such a case, the information in the records which follow shall be logged.

Zero or more of the following records shall be recorded to reflect the information required to represent the high-performance file's characteristics.

BP	Field Name	L	Content	
1–10	Flag	10	`HIGH-PERF:`	
11–	Parameter Specification	--	al-characters	1

Other File Types

For other, implementation-defined, file types, additional information records may be provided as specified below. If the File Size in the first FIF record is nonzero, and if the implementation does not recognize the file type, the subsequent FDF shall be restored as an ordinary file.

NOTE: This implies that the File Size field of the FDF must be accurate for all file types provided as an extension.

BP	Field Name	L	Content	
1–10	Flag	10	a-characters	
11–	Parameter Specification	--	unspecified	1

The Flag may be any combination of a-characters except those that are specified by this standard to appear as the first ten octets of a FIF record. The remaining characters shall either be al-characters or in the alternate character set. If the record type is not recognized by the implementation, it shall be logged;

translation from the alternate character set shall not occur when a record is logged.

4.48.7.6.5 Security Records 1

Following the last file-type specific record, additional records of the following format may be recorded.

HLJ : Help from 1003.6 is solicited.

BP	Field Name	L	Content	
1−10	Flag	10	SECURITY:<space>	2
11−	Security Specification	--	unspecified	1

On systems that do not implement security extensions, these records may be ignored or logged. 2

4.48.7.6.6 Other FIF Information

For all file types, additional information records may be provided as specified below.

BP	Field Name	L	Content	
1−10	Flag	10	a-characters	
11−	Parameter Specification	--	unspecified	1

The Flag may be any combination of a-characters except those that are specified by this standard to appear as the first ten octets of a FIF record. The remaining characters shall either be al-characters or in the alternate character set. If the record type is not recognized by the implementation, it shall be logged; translation from the alternate character set shall not occur when a record is logged.

4.48.7.6.7 The FIF EOF 1 Label

The EOF 1 Record shall be identical to the HDR 1 record recorded for the FIF, except in the following fields.

BP	Field Name	L	P	Content
1−3	Label Identifier	3		EOF
55−60	Block Count	6		<space>, digits

4.48.7.6.7.1 Block Count

This field shall be as specified in ISO 1001 {9}. If the actual block count exceeds the maximum value of a six-digit decimal number, it shall be recorded modulo 2
1 000 000. 2

4.48.7.6.8 The FIF EOF 2−9 and UTL 1−9 Labels

These records are optional, but if present shall be recorded in accordance with ISO 1001. They shall be ignored on input, and it is implementation defined as to 1
whether they are written in output. 1

4.48.7.7 The File Data File

The file data file is optional, and when present shall contain the data contained in the file. If this file is omitted when it is expected, an empty file of the specified type shall be created, and an error logged. If a file of this type is found when it is not expected, the name from the HDR 1 record shall be used as the name of a file containing the data found in the FDF, in an implementation-defined directory. An error shall be logged.

4.48.7.7.1 The FDF HDR 1 Label

BP	Field Name	L	P	Content	
1–3	Label Identifier	3		`HDR`	
4	Label Number	1		`1`	
5–21	File Identifier	17	*	a-characters	
22–27	File Set Identifier	6		a-characters	
28–31	File Section Number	4		digits	
32–35	File Sequence Number	4		digits	
36–39	Generation Number	4		digits	
40–41	Generation Version Number	2		digits	
42–47	Creation Date	6	*	See 4.48.7.7.1.7	1
48–53	Expiration Date	6	*	See 4.48.7.7.1.8	3
54	File Accessibility	1		a-characters	
55–60	Block Count	6		`000000`	1
61–73	Implementation Identification	13	*	`<space>`s	3
74–78	POSIX Identification	5	*	`POSIX`	3
79	POSIX Version	1	*	`1`	3
80	FDF Identifier	1	*	`D`	3
74–80	(Reserved for future POSIX standardization)	7		`<space>`s	

4.48.7.7.1.1 File Identifier

This shall be the name of the FDF file. It shall be derived from the name of the POSIX file being recorded as the filename (not pathname) of the file being recorded, folded to a-characters.

4.48.7.7.1.2 File Set Identifier

This field shall be ignored.

4.48.7.7.1.3 File Section Number

This field shall be as specified in ISO 1001 {9}.

4.48.7.7.1.4 File Sequence Number

This field shall be ignored.

4.48.7.7.1.5 Generation Number

This field shall be ignored.

4.48.7.7.1.6 Generation Version Number

This field shall be ignored.

4.48.7.7.1.7 Creation Date

This shall be the representation of the POSIX.1 {8} *st_ctime* field of the file converted to a Julian day, according to the time zone active at the time `pax` was run. 1

4.48.7.7.1.8 Expiration Date

This shall be the same value as the creation date; see 4.48.7.7.1.7. 3

4.48.7.7.1.9 File Accessibility

HLJ : Help from 1003.6 is solicited.

The content of this field may be any permissible value; `pax` shall ignore this field. 1

4.48.7.7.1.10 Block Count

This field shall be as specified in ISO 1001 {9}.

4.48.7.7.2 Second FDF File Header Label (HDR 2)

BP	Field Name	L	P	Content	
1–3	Label Identifier	3		`HDR`	
4	Label Number	1		`2`	
5	Record Format	1	*	`F`	
6–10	Block Length	5	*	`16384`	1
11–15	Record Length	5	*	`00000`	
16–20	Standards Body	5	*	a-characters	
21–25	Number	5	*	a-characters	
26–30	Variant	5	*	a-characters	
31–34	Revision	4	*	a-characters	
35–50	(Reserved for Implementation)	16		not specified	1
51–52	Offset Length	2		`00`	2
53–80	(Reserved for Standard)	28		`<space>`s	2

4.48.7.7.2.1 Standards Body, Number, Variant, Revision

These fields record the same information, and are encoded in the same way, as the corresponding fields in the VOL2 Header Label. They specify the alternate character set to be used in the FDF file for those fields permitted to be in the alternate character set. If all these fields are spaces, instead of indicating ISO/IEC 646 {1} IRV, the alternate character set specified in the VOL2 label shall be used. If the Standards Body field is ‘`BINARY`’’, the data recorded shall not be translated from the alternate character set.

4.48.7.7.2.2 Record Format

Data is always recorded as one record per block. The data from the file is recorded exactly as it comes from the file.

4.48.7.7.2.3 Block Length

This is the constant 16 384. 1

4.48.7.7.2.4 Record Length

This shall be zeroes.

4.48.7.7.2.5 Offset Length

This field shall contain zeroes. 2

4.48.7.7.3 Additional FDF File Header Labels (HDR 3−9 and UHL ?) 2

These records are not permitted for the FDF. On media with tape marks, if present, they shall be ignored.

4.48.7.7.4 FDF Data

The content of the file shall be recorded in logical blocks of 16 384 octets in length. 2
The last block may be truncated to the actual length.

4.48.7.7.5 The FDF EOF 1 Label

The EOF 1 Record shall be identical to the HDR 1 record recorded for the FIF, except in the following fields.

BP	Field Name	L	P	Content
1−3	Label Identifier	3		`EOF`
55−60	Block Count	6		`<space>`, digits

4.48.7.7.5.1 Block Count

This field shall be as specified in ISO 1001 {9}. If the actual block count exceeds
the maximum value of a six-digit decimal number, it shall be recorded modulo 2
1 000 000. 2

4.48.7.7.6 The FDF EOF 2−9 and UTL 1−9 Labels

These records are optional, but if present shall be recorded in accordance with
ISO 1001 {9}. They shall be ignored on input, and it is implementation defined as 1
to whether they are written in output. 1

4.48.7.7.7 Other labels

Other labels (including the EOV labels) shall be written in accordance with ISO 1001 {9}.

4.48.7.8 Rationale — pax Interchange Format. *(This subclause is not a part of P1003.2b)*

This is a data interchange format for POSIX that meets the requirements of interchange that were discussed in the debates on whether the USTAR or CPIO formats were the appropriate vehicles for such interchange. It is also intended to meet the known additional requirements developed since those debates, and be extensible, in an upward and downward compatible way, to allow future file types. Additionally, it meets the requirements of ISO 1001 {9}, which is equivalent to ANSI X3.27-1987, when certain additional constraints are met. (X3.27 is an old standard, having been around in some form since at least the late 1960s. The 1987 revision appears to be the fourth revision.) This allows a minimal, but useful, level of interchange to systems conforming to ISO 1001 but not to this standard.

Since `pax` (as well as the other POSIX.2 utilities) is not required by the standard 3
to be implemented portably, this interchange format should allow a useful inter- 3
change of data by systems where the tape operations are managed by the operat- 3
ing system as well as by systems where `pax` writes directly to the media. The 3
implementation of `pax` on the former will usually consist of requests for the 3
''header'' and ''user data'' parts of the archive from the system (the system itself 3
reading the headers and providing the information to `pax`). 3

The terminology and concepts taken from ISO 1001 that are needed for understanding this proposal are defined informally here, where the formal definition should be taken from the standard itself. Such terms are flagged the first time they are used with ''(1001),''and excerpts from that standard are quoted.

In a few instances, definitions taken directly from ISO 1001 {9} are included in this rationale.

This format is intended for interchange, not for backup on a single (family of) systems. It is not as densely packed as might be obtained for backup, and contains information as coded characters that for backup could be coded in binary. The whole issue of identification of character sets is probably unnecessary for backup.

Note that there is no requirement on the format of the log, although constraints on the content are mentioned. The log is intended for human use to recover from problems found during restoring the archive.

''Octet'' is chosen both to emphasize the 8-bit nature of this interface, and to avoid
confusion with the meaning of ''byte''as used in most of the rest of this standard. 1

The requirements on restoring from an archive are slightly different from the historical wording, allowing for nonmonolithic privilege to bring forward as much as possible. In particular, attributes such as ''high performance file'' might be broadly but not universally granted while set-user- ID or *chown* () might be much more restricted. There is no implication in this standard that the security information be honored after it is restored to the file hierarchy, in spite of what might be improperly inferred by the silence on that topic. That is a topic for another standard.

If the implementation can make use of further extensions in label records (or if they are automatically generated by some systems) it is permissible use them with an option. This may prove particularly useful for tapes generated on ‘‘label-smart’’ systems.

The character set folding is not ideal, but to be consistent with ISO 1001 {9}, it is required for fields appearing in the header. Note that such fields are only meaningful when the format is not read on a POSIX system or in the case of error, so no crucial data is lost.

Requiring tape marks first of all is required by ISO 1001 {9} for the tape formats above. It also makes recovering a damaged tape much easier. Because media such as regular files do not have the ability to record such things, they are not strictly required for other media, but they are recommended where it is possible.

Note also that this implies that for tape and tape-like devices, a label is identically a block (other than possible padding in the block). This is also intended here. For devices that do not have a block structure like this, then a stream model applies.

ISO 1001 {9} says that there is either only one file section, or if the file will not fit on the current volume (either because it ran out of space along the way, or because the file is too big for one volume) that the second section follow the first on the next volume, and that no other files intervene until the whole file is recorded. (If the file is huge, there may be many sections, but each must use the whole volume until the last.) Because tape marks are ‘‘out of band,’’ it is safe to split data across several volumes like this.

The first form of recording on media other than magnetic tape is specifically addressing either stream devices or recording on an ‘‘ordinary file.’’ The second addresses devices such as sectored disks. Note that without a tape mark or equivalent out-of-band indicator, when dealing with files that span multiple physical volumes, that it is impossible to tell where the data ends and any label records begin. When tape marks are present, this is feasible, and labels are required.

Implementors should note that implementations that refuse to allow writing at least label records after the end-of-tape reflector spot is detected may not be able to conform to this standard (or to ISO 1001 {9}). Historical UNIX systems have not handled the detection of the reflector spot consistent with historical usage in other ‘‘tape-smart’’ systems. Implementations must be able to read data after the end-of-tape reflector spot because other implementations may have written past that point. An implementation is free to back up and write the end-of-tape information after detecting the reflector spot, while holding the previously written information in memory for the next volume. However, given the requirement to read past the reflector spot, that may not be worthwhile to implement.

HLJ : These following paragraphs on multivolume archives need work. 1

Although it is not required by POSIX.1, implementations of the format-reading and -creating utility, upon reading logical end-of-file, may check to see if an error channel is open to a controlling terminal. The utility then produces a message requesting a new medium to be made available. The utility waits for a new medium to be made available by attempting to read a message to restart from the controlling terminal. In all cases, the communication with the controlling terminal is performed in an implementation-defined manner.

The handling of multivolume archives historically has been inconsistent. The 1988 and 1990 versions of POSIX.1 attempted to clarify this in rationale similar to that below. For this new format, the issues of multivolume archives are addressed more precisely. However, for magnetic tape, this needs further clarification because ISO 1001 presumes behavior that many historical systems similar to POSIX have not performed properly in the past.

POSIX.1 is intended to be interpreted such that each byte of the format is represented on the media exactly once.

In some current implementations, it is not deterministic whether encountering the end-of-medium reflector foil on magnetic tape during a write will yield an error during a subsequent *read* () of that record, or even if that record is actually recorded on the tape. It is also possible that *read* () will encounter the end-of-medium when end-of-medium was not encountered when the data was written. This has to do with conditions where the end of (magnetic) record is in such a position that the reflector foil is on the verge of being detected by the sensor and is detected (nondeterministically) during one operation and not on a later one.

An implementation of the format-creating utility must assure when it writes a record that the data appears on the tape exactly once, and is properly followed by the trailer labels required by ISO 1001. This implies that the program and the tape driver work in concert. An implementation of the format-reading utility must assure that an error in a boundary condition described above will not cause loss of data, and that trailer labels will be properly interpreted.

The general consensus was that the following would be considered as correct operation of a tape driver when end-of-medium is detected:

(1) During writing, either:

(a) The record where the reflector spot was detected is backspaced over by the driver so that the trailing tape mark and labels that must be written will overwrite the old record. (It should be noted that in most cases, the trailing labels will be much longer than the original record due to the number of interrecord gaps required.) Writing the labels and tape marks should not yield an end-of-medium condition.

(b) The condition is reported as an error on the *write* () following the one where the end-of-medium is detected (the one where the end-of-medium is actually detected completing successfully). No data will be actually transferred on the *write* () reporting the error, but label records can be written. Writing the tape mark, and writing any subsequent records, should not yield any end-of-medium conditions.

(2) During reading, the end-of-medium indicator is simply ignored, presuming that (end-of-file) labels will be recorded on the magnetic medium and that the reflector foil was advisory only to the *write* ().

Systems where these conditions are not met by the tape driver should assure that the format-creating and -reading utilities assure proper representation and interpretations of the files on the media in a way consistent with the above recommendations.

The typical failures on systems that do not meet the above conditions are either:

(1) To leave the record written when the end-of-medium is encountered on the tape, but to report that it was not written, and possibly not write the trailing labels. The format-creating utility would then rewrite the failed record on the next volume. The format-reading utility could see the record twice if the end-of-medium is not sensed during the read operations.

(2) The *write* () occurs uneventfully, but the *read* () senses the error and does not actually see the data, causing a record to be omitted, and trailing labels are not seen.

Nothing in POSIX.1 requires that end-of-medium be determined by anything on the medium itself (for example, a predetermined maximum size would be an acceptable solution for the format-creating utility). The format-reading utility must be able to *read* () tapes written by machines that do use the whole medium, however.

On media where end-of-medium and end-of-file are reliably coincident, such as disks, end-of-medium and end-of-file can be treated as synonyms.

Note that partial physical records [corresponding to a single *write* ()] can be written on some media, but that only full physical records will actually be written to magnetic tape, given the manner in which the tape operates.

There is wording to specifically to disallow using the ‘‘no tape mark’’ option on magnetic tape, even by agreement, and have it considered as conforming to this standard.

A tape that is a single volume looks (roughly) like this in ISO 1001 {9}:

```
Beginning of Volume Label Group. (VOL? and UVL?)
(
    Beginning of File Label Group. (HDR? and UHL?)
    tape mark
    file data
    tape mark
    End of File Label Group. (EOF? and UTL?)
    tape mark
)*
tape mark
```

NOTE: ('?' and '*' are used in the sense of regular expressions.)

When the file data spans the end of volume, the End of File Label Group becomes an ''End of Section Label Group'' (EOV? and UTL?), and no further data can be recorded on the volume. The next segment of the file must be at the beginning of the next volume, and there are requirements about the consistency of many of the fields between the File Header Label Group for each Section.

ISO 1001 {9}: A sequence of one or more labels of the same type, recorded in consecutive blocks shall be a *label set* of that type. All labels in a set shall be numbered consecutively starting from 1, except those labels in User file Header and User File Trailer Label Sets.

ISO 1001 {9}: The label in User File Header and User File Trailer Label Sets may be identified in any order and may contain duplicate identifiers within a set.

Note that this format yields a double tape mark at the end of the tape. Obviously, in the case of other media, the single volume requirement assures that an End of Section Label Group would never be used.

A tape that is a single volume will look (roughly) like this in the **pax** Data Interchange Format: 1 1

```
Beginning of Volume Label Group (VOL 1, VOL ?, and UVL ?)                1
(                                                                        1
    File Information File Label Group. (HDR 1, HDR 2, and UHL ?)         2
    tape mark (optional, based on media)                                 1
    File Information File data                                           1
    (                                                                    1
                4-digit record size                                      1
                FIF Filename Record                                      1
                4-digit record size                                      1
                FIF Owner Record                                         1
                4-digit record size                                      1
                FIF Group Record                                         1
                4-digit record size                                      1
                FIF Type Dependent Information                           1
                Optional records                                         1
                (                                                        1
                    4-digit record size                                  1
                    FIF Security record (optional)                       1
                    4-digit record size                                  1
                    FIF Information record (optional)                    1
                )                                                        1
    )                                                                    1
    tape mark (optional)                                                 1
    End of File Information File Label Group (EOF 1, EOF ?, and UTL ?)   1
    tape mark (optional)                                                 1
    File Data File Label Group (HDR 1, HDR 2) (if data expected,         1
                else no File Data File)                                  1
    tape mark (optional)                                                 1
    Data in 16384-octet ``logical'' blocks                               2
    tape mark (optional)                                                 1
    End of File Data File Label Group (EOF 1, EOF ?, and UTL ?)          1
    (becomes End Of Volume Group (EOV ? and UTL ?) at end of             1
                Volume or Section)                                       1
    tape mark (optional)                                                 1
)*                                                                       1
tape mark (optional)                                                     1
```

NOTE: ('?' and '*' are used in the sense of shell pattern matching.) 1

It is possible that some implementations will provide a mechanism for setting the Volume Identifier field independent from the archiver, possibly as part of a more general handling of labeled tapes. Note that the format creating utility may have to deal with an interface that assures that the volume identification is retained on the tape when it is rewritten. The POSIX Identifier and version fields are not intended to be required for general labeled tapes.

VOL2 is not specified at all beyond its possible existence in ISO 1001 {9}. Additional labeling information may be stored there.

Only a limited number of character set standards can actually be permitted for maximal interchange. Any character set is of course possible by prior agreement. It has been suggested that EBCDIC be listed. It is not clear whether it is a formal standard; it is omitted. Formal standards, and then only those with reasonably large followings, can be included here, simply as a matter of practicality.

The notation used for an ISO/IEC 646 {1} National Usage variant is appropriate material for a National Profile.

The requirement that translation be able to be suppressed permits mixing binary and character data in a single archive, extracting both, and then translating only the character files. (The character set 'BINARY'' in the FDF is similar; this permits such data to be extracted in the absence of the preplanning needed to use BINARY.) (The `tr` utility will not do because not all the translations are 1:1.) The utility could be *popen* ()-ed to simplify the utility implementation.

ISO 1001 {9} contains the following note: The File Identifier field of a file within a file set is permitted to be the same as that of other files in the file set. Note that this is not the pathname of the file, but rather again exists for the purposes of non-POSIX systems.

ISO 1001 {9}: The File Section Number field shall specify the ordinal number of the file section as a four-digit decimal number. The characters in this field shall be digits.

The format of dates in the Creation and Expiration Date fields means nothing to POSIX, but might be useful to other systems. In ISO 1001 {9} they are Julian dates, accurate only to the day. The coding of the Expiration Date is taken to mean obsolete data by ISO 1001 {9}. This value was selected because certain 2
operating systems prevent overwriting of the tape if the date has not past. 2

ISO 1001 {9}: The Block Count field shall specify a constant value. The characters in this field shall be zeroes.

(Block count is used on the EOF 1 record, which is the same format as HDR 1.)

The Record Format field allows for several long records (such as the pathname) without introducing any complexity beyond that already in ISO 1001.

Note that the four-octet length field is explicitly required even when the record is put on nontape media. In addition to specifying length, they also serve to separate header labels from the content.

Headers can be distinguished from the data because the first record of the actual file will be a variable-length record, which must begin with four digits.

Links are recorded in this fashion described here because a link can be to any file type. It is desirable in general to be able to selectively restore part of an archive, and restore all the files completely. If the data is not associated with each link, it is not possible to do this. However, the data associated with a file can be large, and when selective restoration is not needed, this can be a significant burden.

The archive is structured so that files that have no associated data can always be restored by the name of any linkname of any link, and the user may choose whether data is recorded with each instance of a file that contains data. Although not required of an implementation of `pax`, the format permits mixing of both types of links in a single archive; this can be done for special needs, and `pax` is expected to properly interpret such archives on input.

Note that device/i-number labeling of files is not carried forward from `ustar` and `cpio`; rather, it works strictly on a symbolic name basis. This is intentional because device/i-number links could break if files span file systems that did not used to do so, and because to repair problems it is more useful to have both names.

The file protection modes are those conventionally used by the `ls` utility. This is extended beyond the usage in 1003.2 to support the ‘‘shared text’’ or ‘‘sticky’’ bit. It is intended that the conformance document should not document anything beyond the existence of and support of such a mode. Further extensions are expected to these bits, particularly with overloading the set-user- ID and set-group- ID flags.

Creation time (actually i-node modification time) is for information only, as it is not possible to effectively change it portably. Nothing is intended to prevent a nonportable implementation of the utility from restoring the value. Even though implementation capabilities are moving toward the necessity of finer granularities than seconds, the times here are in seconds since the Epoch since POSIX.1 {8} states them that way. The storage space allocated takes into account that this may change.

Note that the format used for a file name allows for slightly less than 10 000 octets of pathname. This is well above any current practical limit known.

There is a potential issue of name registration for systems, but it is unlikely to be a problem as vendors do try to keep their systems clearly identifiable. Note that this permits someone to write a special file on one system and restore it on an incompatible system by interpreting the meaning in some way.

Delaying restoration of the file ownership and permissions for a directory assures that `pax` will be able to continue to write entries in the directory. The behavior of the file size field is implementation defined to permit, but not endorse, a current USTAR behavior.

The explicit permission for extension through the use of additional FIF records is included because historically archive formats have not been easily extended to allow new information. It is expected that both implementation specific extensions and future standards will use this mechanism. The Flag field was chosen to be sufficiently long to allow such extensions without fear of name conflict. Because it is a known extension currently under development, `SECURITY:` is explicitly reserved.

ISO 1001 {9}: Within a Labeled-Sequence the content of the fields of this label shall be identical with the contents of the corresponding fields in the First File Header Label, except for the following fields.

ISO 1001 {9}: ‘‘[The Block Count] field shall specify, as a six-digit decimal number, the number of blocks in which the file section is recorded.’’

If an unexpected FDF is found, it is intended that the data is put where it can be examined later; the only candidate name (other than a synthetic one) is the one from the HDR 1. The directory should either be the current working directory, or a specified ‘‘errors’’ directory if the utility provides one.

The description of record format seems the best approximation to POSIX ’s 1
unstructured files. No attempt to organize this for non- POSIX systems is made for this archive format.

Because of the unstructured nature of POSIX files, the maximum record length is not specified. In ISO 1001 {9}, this is zero, indicating unspecified length.

Because there is no way to tell how many HDR 3-9 and UHL ? labels there are in 2
the absence of tape marks, they are not permitted because it becomes impossible to detect the start of the actual data. If they are present on magnetic tape anyway, they are ignored simply to allow fail-soft when tapes approximately in this format are written on some systems.

Although the description of FDF Data states that the last block may be truncated to the actual length, note the statements in General Requirements about padding.

4.48.8 Extended Description — `cpio` Interchange Format

HLJ : This is the text adopted from 1003.1 LIS Draft 1. It has been left generally unmodified as a working group exercise.

The octet-oriented `cpio` archive format shall be a series of entries, each comprising a header that describes the file, the name of the file, and then the contents of the file.

An archive may be recorded as a series of fixed-size blocks of octets. This blocking shall be used only to make physical I/O more efficient. The last group of blocks always shall be at the full size.

For the octet-oriented `cpio` archive format, the individual entry information shall be in the order indicated and described by Table 4-100.

Table 4-100 – Octet-Oriented `cpio` Archive Entry

Field Name	Length (in octets)	Interpreted as
Header		
c_magic	6	Octal number
c_dev	6	Octal number
c_ino	6	Octal number
c_mode	6	Octal number
c_uid	6	Octal number
c_gid	6	Octal number
c_nlink	6	Octal number
c_rdev	6	Octal number
c_mtime	11	Octal number
c_namesize	6	Octal number
c_filesize	11	Octal number

Field Name	Length	Interpreted as
File Name		
c_name	*c_namesize*	Pathname string

Field Name	Length	Interpreted as
File Data		
c_filedata	*c_filesize*	Data

4.48.8.1 `cpio` Header

For each file in the archive, a header as defined previously shall be written. The information in the header fields shall be written as streams of ISO/IEC 646 {1} characters interpreted as octal numbers. The octal numbers shall be extended to the necessary length by appending ISO/IEC 646 {1} IRV zeros at the most-significant-digit end of the number; the result is written to the stream of octets most-significant-digit first. The fields shall be interpreted as follows:

c_magic Shall identify the archive as being a transportable archive by containing the identifying value `"070707"`.

c_dev
c_ino Shall contain values that uniquely identify the file within the archive (i.e., no files shall contain the same pair of *c_dev* and *c_ino* values unless they are links to the same file). The values shall be determined in an unspecified manner.

c_mode Shall represent the following values of the file_mode_type:

`04 000`	*set_user_id_on_exec*
`02 000`	*set_group_id_on_exec*
`01 000`	*<reserved>*
`00 400`	*owner_read_permission*
`00 200`	*owner_write_permission*
`00 100`	*owner_exec_or_search_permission*
`00 040`	*group_read_permission*
`00 020`	*group_write_permission*
`00 010`	*group_exec_or_search_permission*
`00 004`	*other_read_permission*
`00 002`	*other_write_permission*
`00 001`	*other_exec_or_search_permission*

and the following values from the file_type_type:

`040 000`	Directory.
`010 000`	FIFO.
`0100 000`	Regular file.
`060 000`	Block special file.
`020 000`	Character special file.
`0110 000`	*<reserved>* .
`0120 000`	*<reserved>* .
`0140 000`	*<reserved>* .

Directories, FIFO s, and regular files shall be supported on a system conforming to this standard; additional values defined previously are reserved for compatibility with existing systems. Additional file types may be supported; however, such files should not be written on archives intended for transport to portable systems.

c_uid Shall contain the user ID of the owner.

c_gid Shall contain the group ID of the group.

c_nlink Shall contain the number of links referencing the file at the time the archive was created.

c_rdev Shall contain implementation-defined information for character or block special files.

c_mtime Shall contain the latest time of modification of the file at the time the archive was created.

c_namesize Shall contain the length of the pathname, including the terminating NUL character.

c_filesize Shall contain the length of the file in octets. This shall be the length of the data section following the header structure.

4.48.8.2 `cpio` File Name

The *c_name* field shall contain the pathname of the file. The length of this field in octets shall be the value of *c_namesize*. If a file name is found on the medium that would create an invalid pathname, the implementation shall define if the data from the file is stored on the file hierarchy and under what name it is stored.

All characters shall be represented in ISO/IEC 646 {1} IRV. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside the portable filename character set in names for files, users, and groups, one or more implementation-defined encodings of these characters shall be provided for interchange purposes. However, the format-reading utility shall never create file names on the local system that cannot be accessed via the procedures described previously in this standard. If a file name is found on the medium that would create an invalid file name, the implementation shall define if the data from the file is stored on the local file system and under what name it is stored. A format-reading utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

4.48.8.3 `cpio` File Data

Following *c_name*, there shall be *c_filesize* octets of data. Interpretation of such data shall occur in a manner dependent on the file. If *c_filesize* is zero, no data shall be contained in *c_filedata*.

When restoring from an archive:

— If the user does not have the appropriate privilege to create a file of the specified type, the format-interpreting utility shall ignore the entry and issue an error to the standard error output.

— Only regular files have data to be restored. Presuming a regular file meets any selection criteria that might be imposed on the format-reading utility by the user, such data shall be restored.

— If a user does not have appropriate privilege to set a particular mode flag, the flag shall be ignored. Some of the mode flags in the archive format are not mentioned elsewhere in this standard. If the implementation does not support those flags, they may be ignored.

4.48.8.4 `cpio` Special Entries

FIFO special files, directories, and the trailer shall be recorded with *c_filesize* equal to zero. For other special files, *c_filesize* is unspecified by this standard. The header for the next file entry in the archive shall be written directly after the last octet of the file entry preceding it. A header denoting the file name ‘`TRAILER!!!`’’ shall indicate the end of the archive; the contents of octets in the last block of the archive following such a header are undefined.

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

The part about appropriate privilege in `cpio` refers to an error on standard output; `tar` does not say that.

The model for this format was the historical System V `cpio −c` data interchange format. This model documents the portable version of `cpio` format and not the binary version. It has the flexibility to transfer data of any type described within the POSIX.1 standard, yet is extensible to transfer data types specific to extensions beyond POSIX.1 (e.g., symbolic links or contiguous files). Because it describes existing practice, there is no question of maintaining upward compatibility.

This subclause does not standardize behavior for the utility when the file type is not understood or supported. It is useful for the utility to report to the user whatever action is taken in this case, though POSIX.1 neither requires nor recommends this.

4.48.8.5.1 `cpio` Header

There has been some concern that the size of the *c_ino* field of the header is too small to handle those systems that have very large i-node numbers. However, the *c_ino* field in the header is used strictly as a hard link resolution mechanism for archives. It is not necessarily the same value as the i-node number of the file in the location from which that file is extracted.

The name *c_magic* is based on historical usage.

4.48.8.5.2 `cpio` File Name

For most historical implementations of the `cpio` utility, {PATH_MAX} octets can be used to describe the pathname without the addition of any other header fields (the NUL character would be included in this count). {PATH_MAX} is the minimum value for pathname size, documented as 256 bytes in Section 2. However, an implementation may use *c_namesize* to determine the exact length of the pathname. With the current description of the `cpio` header, this pathname size can be as large as a number that is described in six octal digits.

Three values are documented under the *c_mode* field values to provide for extensibility for known file types:

`0110 000` Reserved for contiguous files. The implementation may treat the rest of the information for this archive like a regular file. If this file type is undefined, the implementation may create the file as a regular file.

`0120 000` Reserved for files with symbolic links. The implementation may store the link name within the data portion of the file. If this type is undefined, the implementation may not know how to link this file or be able to understand the data section. The implementation may decide to ignore this file type and output a warning message.

`0140 000` Reserved for sockets. If this type is undefined on the target system, the implementation may decide to ignore this file type and output a warning message.

This provides for extensibility of the `cpio` format while allowing for the ability to read old archives. Files of an unknown type may be read as ‘‘regular files’’ on some implementations. On a system that does not support extended file types, the format-interpreting utility should do the best it can with the file and go on to the next.

4.48.9 Extended Description — `ustar` Interchange Format

HLJ : This is the text adopted from 1003.1 LIS Draft 1. It has been left generally 1
unmodified as a working group exercise. 1

A `ustar` archive tape or file shall contain a series of blocks. Each block shall be a fixed-size block of 512 octets (see below). Although this format may be thought of as being stored on 9-track industry-standard 12,7 mm (0,5 in) magnetic tape, other types of transportable media are not excluded. Each file archived shall be represented by a header block that describes the file, followed by zero or more blocks that give the contents of the file. At the end of the archive file there shall be two blocks filled with binary zeroes, interpreted as an end-of-archive indicator.

The blocks may be grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the application utility creating the archive file) may be written with a single write_a_file operation. On magnetic tape, the result of this write shall be a single tape record. The last group of blocks always shall be at the full size, so blocks after the two zero blocks may contain undefined data.

The header block shall be structured as shown in Table 4-101. All lengths and offsets are in decimal.

Table 4-101 – `tar` Header Block 1

Field Name	Octet Offset	Length (in octets)	
name	0	100	1
mode	100	8	1
uid	108	8	1
gid	116	8	1
size	124	12	1
mtime	136	12	1
chksum	148	8	1
typeflag	156	1	1
linkname	157	100	1
magic	257	6	1
version	263	2	1
uname	265	32	1
gname	297	32	1
devmajor	329	8	1
devminor	337	8	1
prefix	345	155	1

All characters in the header block shall be represented in the coded character set of ISO/IEC 646 {1}. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside the portable filename character set in names for files, users, and groups, one or more implementation-defined encodings of these charac-
ters shall be provided for interchange purposes. However, `pax` shall never create 1
file names on the local system that cannot be accessed via the procedures 1

described previously in this standard. If a file name is found on the medium that would create an invalid file name, the implementation shall define if the data from the file is stored on the file hierarchy and under what name it is stored. A format-reading utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

Each field within the header block shall be contiguous; that is, there shall be no padding used. Each character on the archive medium shall be stored contiguously.

The fields *magic*, *uname*, and *gname* shall be character strings each terminated by a NUL character. The fields *name*, *linkname*, and *prefix* shall be NUL-terminated character strings except when all characters in the array contain non-NUL characters including the last character. The *version* field shall be two octets containing the characters `"00"` (zero-zero). The *typeflag* shall contain a single character. All other fields shall be leading zero-filled octal numbers using digits from ISO/IEC 646 {1}IRV. Each numeric field shall be terminated by one or more space or NUL characters.

The *name* and the *prefix* fields shall produce the pathname of the file. The hierarchical relationship of the file can be retained by specifying the pathname as a path prefix, and a slash character and filename as the suffix. A new pathname shall be formed, if *prefix* is not an empty string (its first character is not NUL), by concatenating *prefix* (up to the first NUL character), a slash character, and *name*; otherwise, *name* shall be used alone. In either case, *name* shall be terminated at the first NUL character. If *prefix* begins with a NUL character, it shall be ignored. In this manner, pathnames of at most 256 characters can be supported. If a pathname does not fit in the space provided, the format-creating utility shall notify the user of the error, and no attempt shall be made by the format-creating utility to store any part of the file—header or data—on the medium.

The *linkname* field, described below, shall not use the *prefix* to produce a pathname. As such, a *linkname* is limited to 100 characters. If the name does not fit in the space provided, the format-creating utility shall notify the user of the error, and the utility shall not attempt to store the link on the medium.

The *mode* field provides 12 bits encoded in ISO/IEC 646 {1} octal digit representation. The encoded bits shall represent the following values of the file_mode_type:

`04000`	*set_user_id_on_exec*
`02000`	*set_group_id_on_exec*
`01000`	*<reserved>*
`00400`	*owner_read_permission*
`00200`	*owner_write_permission*
`00100`	*owner_exec_or_search_permission*
`00040`	*group_read_permission*
`00020`	*group_write_permission*
`00010`	*group_exec_or_search_permission*
`00004`	*other_read_permission*
`00002`	*other_write_permission*
`00001`	*other_exec_or_search_permission*

When appropriate privilege is required to set one of these mode bits, and the user restoring the files from the archive does not have the appropriate privilege, the mode bits for which the user does not have appropriate privilege shall be ignored. Some of the mode bits in the archive format are not mentioned elsewhere in this standard. If the implementation does not support those bits, they may be ignored.

The *uid* and *gid* fields shall be the user and group ID of the owner and group of the file, respectively.

The *size* field shall be the size of the file in octets. If the *typeflag* field is set to specify a file to be of type `1` (a link) or `2` (reserved for symbolic links), the *size* field shall be specified as zero. If the *typeflag* field is set to specify a file of type `5` (directory), the *size* field shall be interpreted as described under the definition of that record type. No data blocks shall be stored for types `1`, `2`, or `5`. If the *typeflag* field is set to `3` (character special file), `4` (block special file), or `6` (FIFO), the meaning of the *size* field is unspecified by this standard, and no data blocks shall be stored on the medium. Additionally, for `6`, the *size* field shall be ignored when reading. If the *typeflag* field is set to any other value, the number of blocks written following the header shall be (*size* +511)/512, ignoring any fraction in the result of the division.

The *mtime* field shall be the modification time of the file at the time it was archived. It is the ISO/IEC 646 {1} representation of the octal value of the modification time (*time_file_last_modified*) obtained from the get_file_status_by_path procedure.

The *chksum* field shall be the ISO/IEC 646 {1}IRV representation of the octal value of the simple sum of all octets in the header block. Each octet in the header shall be treated as an unsigned value. These values shall be added to an unsigned integer, initialized to zero, the precision of which shall be not less than 17 bits. When calculating the checksum, the *chksum* field shall be treated as if it were all spaces.

The *typeflag* field shall specify the type of file archived. If a particular implementation does not recognize the type, or the user does not have appropriate privilege to create that type, the file shall be extracted as if it were a regular file if the file type is defined to have a meaning for the size field that could cause data blocks to be written on the medium (see the previous description for *size*). If conversion to an regular file occurs, the format-reading utility shall produce an error indicating that the conversion took place. All of the *typeflag* fields shall be coded in ISO/IEC 646 {1}IRV:

`'0'` Represents a regular file. For backward compatibility, a *typeflag* value of binary zero (`'\0'`) should be recognized as meaning a regular file when extracting files from the archive. Archives written with this version of the archive file format shall create regular files with a *typeflag* value of ISO/IEC 646 {1}IRV `'0'`.

`'1'` Represents a file linked to another file, of any type, previously archived. Such files shall be identified by each file having the same device and file serial number. The linked-to name shall be specified in the *linkname* field with a NUL-character terminator if it is less than 100 octets in length.

`'2'` Reserved to represent a link to another file, of any type, whose device or file serial number differs. This is provided for systems that support linked files whose device or file serial numbers differ, and should be treated as a type `'1'` file if this extension does not exist.

`'3'`,`'4'` Represent character special files and block special files respectively. In this case the *devmajor* and *devminor* fields shall contain information defining the device, the format of which is unspecified by this standard. Implementations may map the device specifications to their own local specification or may ignore the entry.

`'5'` Specifies a directory or subdirectory. On systems where disk allocation is performed on a directory basis, the *size* field shall contain the maximum number of octets (which may be rounded to the nearest disk block allocation unit) that the directory may hold. A *size* field of zero shall indicate no such limiting. Systems that do not support limiting in this manner should ignore the *size* field.

`'6'` Specifies a FIFO special file. Note that the archiving of a FIFO file archives the existence of this file and not its contents.

`'7'` Reserved to represent a file to which an implementation has associated some high performance attribute. Implementations without such extensions should treat this file as a regular file (type `'0'`).

`'A'`-`'Z'` The letters `A` through `Z` are reserved for custom implementations. All other values are reserved for specification in future revisions of this standard.

The *magic* field is the specification that this archive was output in this archive format. If this field contains `"ustar"` (the five ISO/IEC 646 {1}IRV characters shown followed by NUL), the *uname* and *gname* fields shall contain the ISO/IEC 646 {1}IRV representation of the owner and group of the file respectively (truncated to fit, if necessary). When the file is restored by a privileged, protection-preserving version of the utility, the password and group files shall be scanned for these names. If found, the user and group IDs contained within these files shall be used rather than the values contained within the *uid* and *gid* fields.

The encoding of the header is designed to be portable across machines.

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

This description reflects numerous enhancements over pre-1988 versions. The goal of these changes was not only to provide the functional enhancements desired, but also to retain compatibility between new and old versions. This compatibility has been retained. Archives written using the old archive format are compatible with the new format. Archives written using this new format may be read by applications designed to use the old format as long as the functional enhancements provided here are not used. This means the user is limited to archiving only regular type files and nonsymbolic links to such files.

Implementors should be aware that the previous file format did not include a mechanism to archive directory type files. For this reason, the convention of using a file name ending with slash was adopted to specify a directory on the archive.

The total size of the name and prefix fields have been set to meet the minimum requirements for {PATH_MAX}. If a pathname will fit within the name field, it is recommended that the pathname be stored there without the use of the prefix field. Although the name field is known to be too small to contain {PATH_MAX} characters, the value was not changed in this version of the archive file format to retain backward compatibility, and instead the *prefix* was introduced. Also, because of the earlier version of the format, there is no way to remove the restriction on the *linkname* field being limited in size to just that of the *name* field.

The *size* field is required to be meaningful in all implementation extensions, although it could be zero. This is required so that the data blocks can always be properly counted.

It is suggested that if device special files need to be represented that cannot be represented in the standard format that one of the extension types (`'A'`-`'Z'`) be used, and that the additional information for the special file be represented as data and be reflected in the size field.

Attempting to restore a special file type, where it is converted to ordinary data and conflicts with an existing file name, need not be specially detected by the utility. If run as an ordinary user, a format-reading utility should not be able to overwrite the entries in, for example, `/dev` in any case (whether the file is converted to another type or not). If run as a privileged user, it should be able to do so, and it would be considered a bug if it did not. The same is true of ordinary data files and similarly named special files; it is impossible to anticipate the user's needs (who could really intend to overwrite the file), so the behavior should be predictable (and thus regular) and rely on the protection system as required.

The values `'2'` and `'7'` in the typeflag field are intended to define how symbolic links and contiguous files can be stored in a `tar` archive. POSIX.1 does not require the symbolic link or contiguous file extensions, but does define a standard way of archiving these files so that all conforming systems can interpret these file types in a meaningful and consistent manner. On a system that does not support extended file types, the format-interpreting utility should do the best it can with the file and go on to the next.

4.51 pwd — Return working directory name

⇒ **4.51.2 pwd Description.** *Replace the entire Description sublcause with:*

> The `pwd` utility shall write to standard output an absolute pathname of the current working directory with all symbolic link components resolved.

Editor's Note: The following rationale will be added to E.4.51, but is kept here with `pwd` *for this draft:*

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

HLJ : My version of the KornShell doesn't do this. Is that correct?

4.53 rm — Remove directory entries

⇒ **4.53.2 rm Description.** *Replace item (2c) with:*

For each entry contained in *file*, other than dot or dot-dot, the four steps listed here [(1)-(4)] shall be taken with the entry as if it were a *file* operand. The rm utility shall not traverse directories by following symbolic links into other parts of the hierarchy, but shall remove the links themselves.

Editor's Note: The following rationale will be added to E.4.53, but is kept here with rm *for this draft:*

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

The rm utility removes symbolic links themselves, not the files they refer to, as a consequence of the dependence on the POSIX.1 {8} *unlink*() functionality, per the Description. When removing hierachies with −r or −R, the prohibition on following symbolic links has to be made explicit.

4.62 `test` — Evaluate expression

⇒ **4.62.3 `test` Operands.** *Add the following primary in the proper sorted order:*

`−h` *file* True if *file* exists and is a symbolic link.

⇒ **4.62.3 `test` Operands.** *Add the following at the end of the primaries list (before the paragraph that begins* ‘‘A primary can be preceded by . . . ’’

With the exception of the `−h` *file* primary, if a *file* argument is a symbolic link, `test` shall evaluate the expression by resolving the symbolic link and using the file referenced by the link.

Section 5: Revisions to User Portability Utilities Option

5.9 du — Estimate file usage

⇒ **5.9.2 du Description.** *Add a new sentence in the first paragraph, following the sentence beginning with* ‘The du utility, by default . . . ’’

When a symbolic link is encountered in the file hierarchy, du shall count and write the name and size of the symbolic link itself (rather than the file referenced by the link) and shall continue with the other files in the directory containing the link, not attempting to follow the link to another portion of the file hierarchy.

5.39 `file` — Determine file type

⇒ **5.14.1 `file` Synopsis.** *Modify the Synopsis to be:*

`file [−h]` [*file* . . .]

⇒ **5.14.2 `file` Description.** *Add a new paragraph at the end of the subclause:*

If *file* is a symbolic link, by default the link shall be resolved and `file` shall test the type of file referenced by the symbolic link.

⇒ **5.14.3 `file` Options.** *Replace the entire Options subclause with:*

`−h` When a symbolic link is encountered, the link shall not be resolved and `file` shall identify the file as a symbolic link. If `−h` is not specified and *file* is a symbolic link that refers to a nonexistent file, `file` shall identify the file as a symbolic link, as if `−h` had been specified.

⇒ **5.14.6.1 `file` Standard Output.** *Insert a new paragraph between the second and third (the one beginning ‘*If the file named . . . *’’)paragraphs:*

If *file* is identified as a symbolic link (see `−h`), the following alternative output format shall be used:

`"%s: %s %s0"`, <*file*>, <*type*>, <*contents of link*>

⇒ **5.14.6.1 `file` Standard Output.** *Add the following entry to the table named* `file` *Output Strings:*

If *file* is a	<*type* > shall contain the string
symbolic link	`symbolic link to`

Section 6: Revisions to Software Development Utilities Option

There are no revisions to Section 6 (yet).

Annex A
(normative)

Revisions to C Language Development Utilities Option

There are no revisions to Annex A (yet).

Annex B
(norm ative)

Revisions to C Language Bindings Option

Editor's Note: This annex is new for Draft 3. It is not further diff-marked. 3

B.2.5 Pathname Variable Values

⇒ **B.2.3 Pathname Variable Values.** *Add a new subclause, B.2.5, Pathname Variable Values, as follows:*

The macros in Table B-100 can be used by the application at execution time to determine which optional facilities relate to specific pathnames. A definition of one of the values from Table B-100 shall be omitted from `<limits.h>` on specific implementations where the corresponding value is equal to or greater than the stated minimum, but where the value can vary depending on the file to which it is applied. The actual value supported for a specific pathname shall be provided by the *pathconf* () function.

Table B-100 – C Pathname Variable Values

Name	Description
_POSIX 2_SYMLINKS	When referring to a directory. the system supports the creation of symbolic links within that directory; for nondirectory files, the meaning of {POSIX 2_SYMLINKS } is undefined.

⇒ **B.10.2 C Binding for Get Numeric-Valued Configurable Variables** *Add the following sentence to the end of the only paragraph in the subclause:*

In addition, the *pathconf* () and *fpathconf* () functions shall support the *name* values in Table B-101, defined in `<unistd.h>`, to provide values for variables in 2.13.3.

Table B-101 – C Bindings for Pathname Configurable Variables

Symbolic Limit	*name* **Value**
{POSIX2_SYMLINKS}	_SC_2_SYMLINKS

Alphabetic Topical Index

E

F

G

H

I

R

S

T

U

V

W