Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

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]

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.

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.

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

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. 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 1 000 000.

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 to whether they are written in output.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 to be implemented portably, this interchange format should allow a useful interchange of data by systems where the tape operations are managed by the operating system as well as by systems where `pax` writes directly to the media. The implementation of `pax` on the former will usually consist of requests for the ''header'' and ''user data'' parts of the archive from the system (the system itself reading the headers and providing the information to `pax`).

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.

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

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

HLJ : These following paragraphs on multivolume archives need work.

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

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

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

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

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 operating systems prevent overwriting of the tape if the date has not past. 2 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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 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 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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

Header		
Field Name	**Length (in octets)**	**Interpreted as**
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

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

File Data		
Field Name	**Length**	**Interpreted as**
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:

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1434 *c_magic* Shall identify the archive as being a transportable archive by con-
1435 taining the identifying value `"070707"`.

1436 *c_dev*
1437 *c_ino* Shall contain values that uniquely identify the file within the
1438 archive (i.e., no files shall contain the same pair of *c_dev* and
1439 *c_ino* values unless they are links to the same file). The values
1440 shall be determined in an unspecified manner.

1441 *c_mode* Shall represent the following values of the file_mode_type:

1442 `04 000` *set_user_id_on_exec*
1443 `02 000` *set_group_id_on_exec*
1444 `01 000` *<reserved>*
1445 `00 400` *owner_read_permission*
1446 `00 200` *owner_write_permission*
1447 `00 100` *owner_exec_or_search_permission*
1448 `00 040` *group_read_permission*
1449 `00 020` *group_write_permission*
1450 `00 010` *group_exec_or_search_permission*
1451 `00 004` *other_read_permission*
1452 `00 002` *other_write_permission*
1453 `00 001` *other_exec_or_search_permission*

1454 and the following values from the file_type_type:

1455 `040 000` Directory.
1456 `010 000` FIFO.
1457 `0100 000` Regular file.
1458 `060 000` Block special file.
1459 `020 000` Character special file.
1460 `0110 000` *<reserved>*.
1461 `0120 000` *<reserved>*.
1462 `0140 000` *<reserved>*.

1463 Directories, FIFO s, and regular files shall be supported on a sys-
1464 tem conforming to this standard; additional values defined previ-
1465 ously are reserved for compatibility with existing systems. Addi-
1466 tional file types may be supported; however, such files should not
1467 be written on archives intended for transport to portable systems.

1468 *c_uid* Shall contain the user ID of the owner.

1469 *c_gid* Shall contain the group ID of the group.

1470 *c_nlink* Shall contain the number of links referencing the file at the time
1471 the archive was created.

1472 *c_rdev* Shall contain implementation-defined information for character
1473 or block special files.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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:

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1551 `0110000` Reserved for contiguous files. The implementation may treat the
1552 rest of the information for this archive like a regular file. If this
1553 file type is undefined, the implementation may create the file as a
1554 regular file.

1555 `0120000` Reserved for files with symbolic links. The implementation may
1556 store the link name within the data portion of the file. If this
1557 type is undefined, the implementation may not know how to link
1558 this file or be able to understand the data section. The implemen-
1559 tation may decide to ignore this file type and output a warning
1560 message.

1561 `0140000` Reserved for sockets. If this type is undefined on the target sys-
1562 tem, the implementation may decide to ignore this file type and
1563 output a warning message.

1564 This provides for extensibility of the `cpio` format while allowing for the ability to
1565 read old archives. Files of an unknown type may be read as ‘‘regular files’’ on
1566 some implementations. On a system that does not support extended file types,
1567 the format-interpreting utility should do the best it can with the file and go on to
1568 the next.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 unmodified as a working group exercise.

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

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

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 characters shall be provided for interchange purposes. However, `pax` shall never create file names on the local system that cannot be accessed via the procedures

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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:

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*

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

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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,

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

Copyright © 1992 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.