Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'pax.1p' command

$ man pax.1p

PAX(1P) POSIX Programmer's Manual PAX(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

pax ? portable archive interchange

SYNOPSIS

pax [-dv] [-c|-n] [-H|-L] [-o options] [-f archive] [-s replstr]... [pattern...]

pax -r[-c|-n] [-dikuv] [-H|-L] [-f archive] [-o options]... [-p string]... [-s replstr]... [pattern...]

pax -w [-dituvX] [-H|-L] [-b blocksize] [[-a] [-f archive]] [-o options]... [-s replstr]... [-x format] [file...]

pax -r -w [-diklntuvX] [-H|-L] [-o options]... [-p string]... [-s replstr]... [file...] directory

DESCRIPTION

The pax utility shall read, write, and write lists of the members of archive files and copy directory hierarchies. A variety of archive for? mats shall be supported; see the -x format option.

The action to be taken depends on the presence of the -r and -w op? tions. The four combinations of -r and -w are referred to as the four

modes of operation: list, read, write, and copy modes, corresponding respectively to the four forms shown in the SYNOPSIS section.

list In list mode (when neither -r nor -w are specified), pax shall write the names of the members of the archive file read from the standard input, with pathnames matching the speci? fied patterns, to standard output. If a named file is of type directory, the file hierarchy rooted at that file shall be listed as well.

read In read mode (when -r is specified, but -w is not), pax shall extract the members of the archive file read from the stan? dard input, with pathnames matching the specified patterns. If an extracted file is of type directory, the file hierarchy rooted at that file shall be extracted as well. The extracted files shall be created performing pathname resolution with the directory in which pax was invoked as the current working directory.

If an attempt is made to extract a directory when the direc? tory already exists, this shall not be considered an error.

If an attempt is made to extract a FIFO when the FIFO already exists, this shall not be considered an error.

The ownership, access, and modification times, and file mode of the restored files are discussed under the -p option.

write In write mode (when -w is specified, but -r is not), pax shall write the contents of the file operands to the standard output in an archive format. If no file operands are speci? fied, a list of files to copy, one per line, shall be read from the standard input and each entry in this list shall be processed as if it had been a file operand on the command line. A file of type directory shall include all of the files in the file hierarchy rooted at the file.

copy In copy mode (when both -r and -w are specified), pax shall copy the file operands to the destination directory.

If no file operands are specified, a list of files to copy,

one per line, shall be read from the standard input. A file of type directory shall include all of the files in the file hierarchy rooted at the file.

The effect of the copy shall be as if the copied files were written to a pax format archive file and then subsequently extracted, except that copying of sockets may be supported even if archiving them in write mode is not supported, and that there may be hard links between the original and the copied files. If the destination directory is a subdirectory of one of the files to be copied, the results are unspeci? fied. If the destination directory is a file of a type not defined by the System Interfaces volume of POSIX.1?2017, the results are implementation-defined; otherwise, it shall be an error for the file named by the directory operand not to ex? ist, not be writable by the user, or not be a file of type directory.

In read or copy modes, if intermediate directories are necessary to ex? tract an archive member, pax shall perform actions equivalent to the mkdir() function defined in the System Interfaces volume of POSIX.1?2017, called with the following arguments:

* The intermediate directory used as the path argument
* The value of the bitwise-inclusive OR of S_IRWXU, S_IRWXG, and S_IRWXO as the mode argument

If any specified pattern or file operands are not matched by at least one file or archive member, pax shall write a diagnostic message to standard error for each one that did not match and exit with a non-zero exit status.

The archive formats described in the EXTENDED DESCRIPTION section shall be automatically detected on input. The default output archive format shall be implementation-defined.

A single archive can span multiple files. The pax utility shall deter? mine, in an implementation-defined manner, what file to read or write as the next file.

If the selected archive format supports the specification of linked files, it shall be an error if these files cannot be linked when the archive is extracted. For archive formats that do not store file con? tents with each name that causes a hard link, if the file that contains the data is not extracted during this pax session, either the data shall be restored from the original file, or a diagnostic message shall be displayed with the name of a file that can be used to extract the data. In traversing directories, pax shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file visited. When it detects an infinite loop, pax shall write a diagnostic message to standard error and shall terminate.

OPTIONS

The pax utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines, except that the order of presentation of the -o, -p, and -s options is significant.

The following options shall be supported:

-r Read an archive file from standard input.

-w Write files to the standard output in the specified archive format.

-a Append files to the end of the archive. It is implementation-defined which devices on the system support appending. Addi? tional file formats unspecified by this volume of POSIX.1?2017 may impose restrictions on appending.

-b blocksize

Block the output at a positive decimal integer number of bytes per write to the archive file. Devices and archive for? mats may impose restrictions on blocking. Blocking shall be automatically determined on input. Conforming applications shall not specify a blocksize value larger than 32256. De? fault blocking when creating archives depends on the archive format. (See the -x option below.)

-c Match all file or archive members except those specified by the pattern or file operands.

-d Cause files of type directory being copied or archived or ar? chive members of type directory being extracted or listed to match only the file or archive member itself and not the file hierarchy rooted at the file.

-f archive

Specify the pathname of the input or output archive, overrid? ing the default standard input (in list or read modes) or standard output (write mode).

-H If a symbolic link referencing a file of type directory is specified on the command line, pax shall archive the file hi? erarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Other? wise, if a symbolic link referencing a file of any other file type which pax can normally archive is specified on the com? mand line, then pax shall archive the file referenced by the link, using the name of the link. The default behavior, when neither -H or -L are specified, shall be to archive the sym? bolic link itself.

-i Interactively rename files or archive members. For each ar? chive member matching a pattern operand or file matching a file operand, a prompt shall be written to the file /dev/tty. The prompt shall contain the name of the file or archive mem? ber, but the format is otherwise unspecified. A line shall then be read from /dev/tty. If this line is blank, the file or archive member shall be skipped. If this line consists of a single period, the file or archive member shall be pro? cessed with no modification to its name. Otherwise, its name shall be replaced with the contents of the line. The pax utility shall immediately exit with a non-zero exit status if end-of-file is encountered when reading a response or if /dev/tty cannot be opened for reading and writing.

The results of extracting a hard link to a file that has been renamed during extraction are unspecified.

-k Prevent the overwriting of existing files.

-l (The letter ell.) In copy mode, hard links shall be made be? tween the source and destination file hierarchies whenever possible. If specified in conjunction with -H or -L, when a symbolic link is encountered, the hard link created in the destination file hierarchy shall be to the file referenced by the symbolic link. If specified when neither -H nor -L is specified, when a symbolic link is encountered, the implemen? tation shall create a hard link to the symbolic link in the source file hierarchy or copy the symbolic link to the desti? nation.

-L If a symbolic link referencing a file of type directory is specified on the command line or encountered during the tra? versal of a file hierarchy, pax shall archive the file hier? archy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Other? wise, if a symbolic link referencing a file of any other file type which pax can normally archive is specified on the com? mand line or encountered during the traversal of a file hier? archy, pax shall archive the file referenced by the link, us? ing the name of the link. The default behavior, when neither -H or -L are specified, shall be to archive the symbolic link itself.

-n Select the first archive member that matches each pattern op? erand. No more than one archive member shall be matched for each pattern (although members of type directory shall still match the file hierarchy rooted at that file).

-o options

Provide information to the implementation to modify the algo? rithm for extracting or writing files. The value of options shall consist of one or more <comma>-separated keywords of the form:

keyword[[:]=value][,keyword[[:]=value], ...]

Some keywords apply only to certain file formats, as indi? cated with each description. Use of keywords that are inap? plicable to the file format being processed produces unde? fined results.

Keywords in the options argument shall be a string that would be a valid portable filename as described in the Base Defini? tions volume of POSIX.1?2017, Section 3.282, Portable File? name Character Set.

Note: Keywords are not expected to be filenames, merely to follow the same character composition rules as portable filenames.

Keywords can be preceded with white space. The value field shall consist of zero or more characters; within value, the application shall precede any literal <comma> with a <back? slash>, which shall be ignored, but preserves the <comma> as part of value. A <comma> as the final character, or a <comma> followed solely by white space as the final charac? ters, in options shall be ignored. Multiple -o options can be specified; if keywords given to these multiple -o options conflict, the keywords and values appearing later in command line sequence shall take precedence and the earlier shall be silently ignored. The following keyword values of options shall be supported for the file formats as indicated:

delete=pattern

(Applicable only to the -x pax format.) When used in write or copy mode, pax shall omit from extended header records that it produces any keywords matching the string pattern. When used in read or list mode, pax shall ignore any keywords matching the string pattern in the extended header records. In both cases, matching shall be performed using the pattern matching notation described in Section 2.13.1, Patterns Matching a Single Character and Section 2.13.2, Patterns Matching Multi?

ple Characters. For example:

-o delete=security.*

would suppress security-related information. See pax Extended Header for extended header record keyword us? age.

When multiple -odelete=pattern options are specified, the patterns shall be additive; all keywords matching the specified string patterns shall be omitted from ex? tended header records that pax produces.

exthdr.name=string

(Applicable only to the -x pax format.) This keyword allows user control over the name that is written into the ustar header blocks for the extended header pro? duced under the circumstances described in pax Header Block. The name shall be the contents of string, after the following character substitutions have been made:

string	Replaced by:
%d	The directory name of the file, equiv? alent to the result of the dirname utility on the translated pathname.
%f	The filename of the file, equivalent to the result of the basename utility on the translated pathname.
%p	The process ID of the pax process.
%%	A '%' character.

Any other '%' characters in string produce undefined results.

If no -o exthdr.name=string is specified, pax shall use the following default value:

%d/PaxHeaders.%p/%f

globexthdr.name=string

(Applicable only to the -x pax format.) When used in write or copy mode with the appropriate options, pax shall create global extended header records with ustar header blocks that will be treated as regular files by previous versions of pax. This keyword allows user control over the name that is written into the ustar header blocks for global extended header records. The name shall be the contents of string, after the follow? ing character substitutions have been made:

string	Replaced by:
%n	An integer that represents the sequence number of the global extended header record in the archive, starting at 1.
%p	The process ID of the pax process.
%%	A '%' character.

Any other '%' characters in string produce undefined results.

If no -o globexthdr.name=string is specified, pax shall use the following default value:

$TMPDIR/GlobalHead.%p.%n

where $TMPDIR represents the value of the TMPDIR envi? ronment variable. If TMPDIR is not set, pax shall use /tmp.

invalid=action

(Applicable only to the -x pax format.) This keyword allows user control over the action pax takes upon en?

countering values in an extended header record that, in read or copy mode, are invalid in the destination hier? archy or, in list mode, cannot be written in the code? set and current locale of the implementation. The fol? lowing are invalid values that shall be recognized by pax:

-- In read or copy mode, a filename or link name that contains character encodings invalid in the desti? nation hierarchy. (For example, the name may con? tain embedded NULs.)

-- In read or copy mode, a filename or link name that is longer than the maximum allowed in the destina? tion hierarchy (for either a pathname component or the entire pathname).

-- In list mode, any character string value (filename, link name, user name, and so on) that cannot be written in the codeset and current locale of the implementation.

The following mutually-exclusive values of the action argument are supported:

binary In write mode, pax shall generate a hdr? charset=BINARY extended header record for each file with a filename, link name, group name, owner name, or any other field in an extended header record that cannot be trans? lated to the UTF?8 codeset, allowing the ar? chive to contain the files with unencoded ex? tended header record values. In read or copy mode, pax shall use the values specified in the header without translation, regardless of whether this may overwrite an existing file with a valid name. In list mode, pax shall behave identically to the bypass action.

bypass In read or copy mode, pax shall bypass the file, causing no change to the destination hierarchy. In list mode, pax shall write all requested valid values for the file, but its method for writing invalid values is unspeci? fied.

rename In read or copy mode, pax shall act as if the -i option were in effect for each file with invalid filename or link name values, allow? ing the user to provide a replacement name interactively. In list mode, pax shall be? have identically to the bypass action.

UTF?8 When used in read, copy, or list mode and a filename, link name, owner name, or any other field in an extended header record cannot be translated from the pax UTF?8 codeset format to the codeset and current locale of the im? plementation, pax shall use the actual UTF?8 encoding for the name. If a hdrcharset ex? tended header record is in effect for this file, the character set specified by that record shall be used instead of UTF?8. If a hdrcharset=BINARY extended header record is in effect for this file, no translation shall be performed.

write In read or copy mode, pax shall write the file, translating the name, regardless of whether this may overwrite an existing file with a valid name. In list mode, pax shall behave identically to the bypass action.

If no -o invalid=option is specified, pax shall act as if -oinvalid=bypass were specified. Any overwriting of existing files that may be allowed by the -oinvalid=

actions shall be subject to permission (-p) and modifi?
cation time (-u) restrictions, and shall be suppressed
if the -k option is also specified.

linkdata

(Applicable only to the -x pax format.) In write mode,
pax shall 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 ar?
chive.

listopt=format

This keyword specifies the output format of the table
of contents produced when the -v option is specified in
list mode. See List Mode Format Specifications. To
avoid ambiguity, the listopt=format shall be the only
or final keyword=value pair in a -o option-argument;
all characters in the remainder of the option-argument
shall be considered part of the format string. When
multiple -olistopt=format options are specified, the
format strings shall be considered a single, concate?
nated string, evaluated in command line order.

times

(Applicable only to the -x pax format.) When used in
write or copy mode, pax shall include atime and mtime
extended header records for each file. See pax Extended
Header File Times.

In addition to these keywords, if the -x pax format is speci?
fied, any of the keywords and values defined in pax Extended
Header, including implementation extensions, can be used in
-o option-arguments, in either of two modes:

keyword=value

When used in write or copy mode, these keyword/value
pairs shall be included at the beginning of the archive
as typeflag g global extended header records. When used

in read or list mode, these keyword/value pairs shall act as if they had been at the beginning of the archive as typeflag g global extended header records.

keyword:=value

When used in write or copy mode, these keyword/value pairs shall be included as records at the beginning of a typeflag x extended header for each file. (This shall be equivalent to the <equals-sign> form except that it creates no typeflag g global extended header records.) When used in read or list mode, these keyword/value pairs shall act as if they were included as records at the end of each extended header; thus, they shall over? ride any global or file-specific extended header record keywords of the same names. For example, in the com? mand:

```
pax -r -o "
gname:=mygroup,
" <archive
```

the group name will be forced to a new value for all files read from the archive.

The precedence of -o keywords over various fields in the ar? chive is described in pax Extended Header Keyword Precedence.

If the -o delete=pattern, -o keyword=value, or -o key? word:=value options are used to override or remove any ex? tended header data needed to find files in an archive (e.g., -o delete=size for a file whose size cannot be represented in a ustar header or -o size=100 for a file whose size is not 100 bytes), the behavior is undefined.

-p string Specify one or more file characteristic options (privileges).

The string option-argument shall be a string specifying file characteristics to be retained or discarded on extraction. The string shall consist of the specification characters a, e, m, o, and p. Other implementation-defined characters can

be included. Multiple characteristics can be concatenated within the same string and multiple -p options can be speci? fied. The meaning of the specification characters are as fol? lows:

a Do not preserve file access times.

e Preserve the user ID, group ID, file mode bits (see the Base Definitions volume of POSIX.1?2017, Section 3.169, File Mode Bits), access time, modification time, and any other implementation-defined file characteristics.

m Do not preserve file modification times.

o Preserve the user ID and group ID.

p Preserve the file mode bits. Other implementation-de? fined file mode attributes may be preserved.

In the preceding list, ``preserve'' indicates that an attri? bute stored in the archive shall be given to the extracted file, subject to the permissions of the invoking process. The access and modification times of the file shall be preserved unless otherwise specified with the -p option or not stored in the archive. All attributes that are not preserved shall be determined as part of the normal file creation action (see Section 1.1.1.4, File Read, Write, and Creation).

If neither the e nor the o specification character is speci? fied, or the user ID and group ID are not preserved for any reason, pax shall not set the S_ISUID and S_ISGID bits of the file mode.

If the preservation of any of these items fails for any rea? son, pax shall write a diagnostic message to standard error. Failure to preserve these items shall affect the final exit status, but shall not cause the extracted file to be deleted.

If file characteristic letters in any of the string option-arguments are duplicated or conflict with each other, the ones given last shall take precedence. For example, if -p eme is specified, file modification times are preserved.

-s replstr

Modify file or archive member names named by pattern or file operands according to the substitution expression replstr, using the syntax of the ed utility. The concepts of ``ad? dress'' and ``line'' are meaningless in the context of the pax utility, and shall not be supplied. The format shall be:

-s /old/new/[gp]

where as in ed, old is a basic regular expression and new can contain an <ampersand>, '\n' (where n is a digit) back-refer? ences, or subexpression matching. The old string shall also be permitted to contain <newline> characters.

Any non-null character can be used as a delimiter ('/' shown here). Multiple -s expressions can be specified; the expres? sions shall be applied in the order specified, terminating with the first successful substitution. The optional trail? ing 'g' is as defined in the ed utility. The optional trail? ing 'p' shall cause successful substitutions to be written to standard error. File or archive member names that substitute to the empty string shall be ignored when reading and writing archives.

-t When reading files from the file system, and if the user has the permissions required by utime() to do so, set the access time of each file read to the access time that it had before being read by pax.

-u Ignore files that are older (having a less recent file modi? fication time) than a pre-existing file or archive member with the same name. In read mode, an archive member with the same name as a file in the file system shall be extracted if the archive member is newer than the file. In write mode, an archive file member with the same name as a file in the file system shall be superseded if the file is newer than the ar? chive member. If -a is also specified, this is accomplished by appending to the archive; otherwise, it is unspecified

whether this is accomplished by actual replacement in the ar?
chive or by appending to the archive. In copy mode, the file
in the destination hierarchy shall be replaced by the file in
the source hierarchy or by a link to the file in the source
hierarchy if the file in the source hierarchy is newer.

-v In list mode, produce a verbose table of contents (see the
STDOUT section). Otherwise, write archive member pathnames
to standard error (see the STDERR section).

-x format Specify the output archive format. The pax utility shall sup?
port the following formats:

cpio The cpio interchange format; see the EXTENDED DE?
SCRIPTION section. The default blocksize for this
format for character special archive files shall be
5120. Implementations shall support all blocksize
values less than or equal to 32256 that are multi?
ples of 512.

pax The pax interchange format; see the EXTENDED DE?
SCRIPTION section. The default blocksize for this
format for character special archive files shall be
5120. Implementations shall support all blocksize
values less than or equal to 32256 that are multi?
ples of 512.

ustar The tar interchange format; see the EXTENDED DE?
SCRIPTION section. The default blocksize for this
format for character special archive files shall be
10240. Implementations shall support all blocksize
values less than or equal to 32256 that are multi?
ples 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 differ?
ent from the existing archive format shall cause pax to exit

immediately with a non-zero exit status.

-X When traversing the file hierarchy specified by a pathname, pax shall not descend into directories that have a different device ID (st_dev; see the System Interfaces volume of POSIX.1?2017, stat()).

Specifying more than one of the mutually-exclusive options -H and -L shall not be considered an error and the last option specified shall determine the behavior of the utility.

The options that operate on the names of files or archive members (-c, -i, -n, -s, -u, and -v) shall interact as follows. In read mode, the archive members shall be selected based on the user-specified pattern operands as modified by the -c, -n, and -u options. Then, any -s and -i options shall modify, in that order, the names of the selected files.

The -v option shall write names resulting from these modifications.

In write mode, the files shall be selected based on the user-specified pathnames as modified by the -n and -u options. Then, any -s and -i op? tions shall modify, in that order, the names of these selected files.

The -v option shall write names resulting from these modifications.

If both the -u and -n options are specified, pax shall not consider a file selected unless it is newer than the file to which it is compared.

List Mode Format Specifications

In list mode with the -o listopt=format option, the format argument shall be applied for each selected file. The pax utility shall append a <newline> to the listopt output for each selected file. The format ar? gument shall be used as the format string described in the Base Defini? tions volume of POSIX.1?2017, Chapter 5, File Format Notation, with the exceptions 1. through 6. defined in the EXTENDED DESCRIPTION section of printf, plus the following exceptions:

7. The sequence (keyword) can occur before a format conversion spec? ifier. The conversion argument is defined by the value of key? word. The implementation shall support the following keywords:

-- Any of the Field Name entries in Table 4-14, ustar Header Block and Table 4-16, Octet-Oriented cpio Archive Entry. The

implementation may support the cpio keywords without the leading c_ in addition to the form required by Table 4-16, Octet-Oriented cpio Archive Entry.

-- Any keyword defined for the extended header in pax Extended Header.

-- Any keyword provided as an implementation-defined extension within the extended header defined in pax Extended Header.

For example, the sequence "%(charset)s" is the string value of the name of the character set in the extended header.

The result of the keyword conversion argument shall be the value from the applicable header field or extended header, without any trailing NULs.

All keyword values used as conversion arguments shall be trans? lated from the UTF?8 encoding (or alternative encoding specified by any hdrcharset extended header record) to the character set appropriate for the local file system, user database, and so on, as applicable.

8. An additional conversion specifier character, T, shall be used to specify time formats. The T conversion specifier character can be preceded by the sequence (keyword=subformat), where subformat is a date format as defined by date operands. The default keyword shall be mtime and the default subformat shall be:

   %b %e %H:%M %Y

9. An additional conversion specifier character, M, shall be used to specify the file mode string as defined in ls Standard Output. If (keyword) is omitted, the mode keyword shall be used. For exam? ple, %.1M writes the single character corresponding to the <en? try type> field of the ls -l command.

10. An additional conversion specifier character, D, shall be used to specify the device for block or special files, if applicable, in an implementation-defined format. If not applicable, and (key? word) is specified, then this conversion shall be equivalent to %(keyword)u. If not applicable, and (keyword) is omitted, then

this conversion shall be equivalent to <space>.

11. An additional conversion specifier character, F, shall be used to specify a pathname. The F conversion character can be preceded by a sequence of <comma>-separated keywords:

(keyword[,keyword] ...)

The values for all the keywords that are non-null shall be con? catenated together, each separated by a '/'. The default shall be (path) if the keyword path is defined; otherwise, the default shall be (prefix,name).

12. An additional conversion specifier character, L, shall be used to specify a symbolic link expansion. If the current file is a sym? bolic link, then %L shall expand to:

"%s -> %s", <value of keyword>, <contents of link>

Otherwise, the %L conversion specification shall be the equiva? lent of %F.

OPERANDS

The following operands shall be supported:

directory The destination directory pathname for copy mode.

file A pathname of a file to be copied or archived.

pattern A pattern matching one or more pathnames of archive members.

A pattern must be given in the name-generating notation of the pattern matching notation in Section 2.13, Pattern Match? ing Notation, including the filename expansion rules in Sec? tion 2.13.3, Patterns Used for Filename Expansion. The de? fault, if no pattern is specified, is to select all members in the archive.

STDIN

In write mode, the standard input shall be used only if no file oper? ands are specified. It shall be a file containing a list of pathnames, each terminated by a <newline> character.

In list and read modes, if -f is not specified, the standard input shall be an archive file.

Otherwise, the standard input shall not be used.

INPUT FILES

The input file named by the archive option-argument, or standard input when the archive is read from there, shall be a file formatted accord? ing to one of the specifications in the EXTENDED DESCRIPTION section or some other implementation-defined format.

The file /dev/tty shall be used to write prompts and read responses.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of pax:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the pattern matching expressions for the pattern operand, the ba? sic regular expression for the -s option, and the extended regular expression defined for the yesexpr locale keyword in the LC_MESSAGES category.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes used in the ex? tended regular expression defined for the yesexpr locale key? word in the LC_MESSAGES category, and pattern matching.

LC_MESSAGES

Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of di? agnostic messages and prompts written to standard error.

LC_TIME Determine the format and contents of date and time strings

when the -v option is specified.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

TMPDIR Determine the pathname that provides part of the default global extended header record file, as described for the -o globexthdr= keyword in the OPTIONS section.

TZ Determine the timezone used to calculate date and time strings when the -v option is specified. If TZ is unset or null, an unspecified default timezone shall be used.

ASYNCHRONOUS EVENTS

Default.

STDOUT

In write mode, if -f is not specified, the standard output shall be the archive formatted according to one of the specifications in the EX? TENDED DESCRIPTION section, or some other implementation-defined format (see -x format).

In list mode, when the -olistopt=format has been specified, the se? lected archive members shall be written to standard output using the format described under List Mode Format Specifications. In list mode without the -olistopt=format option, the table of contents of the se? lected archive members shall be written to standard output using the following format:

"%s\n", <pathname>

If the -v option is specified in list mode, the table of contents of the selected archive members shall be written to standard output using the following formats.

For pathnames representing hard links to previous members of the ar? chive:

"%s == %s\n", <ls -l listing>, <linkname>

For all other pathnames:

"%s\n", <ls -l listing>

where <ls -l listing> shall be the format specified by the ls utility with the -l option. When writing pathnames in this format, it is un?

specified what is written for fields for which the underlying archive format does not have the correct information, although the correct num? ber of <blank>-separated fields shall be written.

In list mode, standard output shall not be buffered more than a path? name (plus any associated information and a <newline> terminator) at a time.

STDERR

If -v is specified in read, write, or copy modes, pax shall write the pathnames it processes to the standard error output using the following format:

"%s\n", <pathname>

These pathnames shall be written as soon as processing is begun on the file or archive member, and shall be flushed to standard error. The trailing <newline>, which shall not be buffered, is written when the file has been read or written.

If the -s option is specified, and the replacement string has a trail? ing 'p', substitutions shall be written to standard error in the fol? lowing format:

"%s >> %s\n", <original pathname>, <new pathname>

In all operating modes of pax, optional messages of unspecified format concerning the input archive format and volume number, the number of files, blocks, volumes, and media parts as well as other diagnostic messages may be written to standard error.

In all formats, for both standard output and standard error, it is un? specified how non-printable characters in pathnames or link names are written.

When using the -xpax archive format, if a filename, link name, group name, owner name, or any other field in an extended header record can? not be translated between the codeset in use for that extended header record and the character set of the current locale, pax shall write a diagnostic message to standard error, shall process the file as de? scribed for the -o invalid= option, and then shall continue processing with the next file.

OUTPUT FILES

In read mode, the extracted output files shall be of the archived file type. In copy mode, the copied output files shall be the type of the file being copied. In either mode, existing files in the destination hierarchy shall be overwritten only when all permission (-p), modifica? tion time (-u), and invalid-value (-oinvalid=) tests allow it.

In write mode, the output file named by the -f option-argument shall be a file formatted according to one of the specifications in the EXTENDED DESCRIPTION section, or some other implementation-defined format.

EXTENDED DESCRIPTION

pax Interchange Format

A pax archive tape or file produced in the -xpax format shall contain a series of blocks. The physical layout of the archive shall be identical to the ustar format described in ustar Interchange Format. Each file archived shall be represented by the following sequence:

* An optional header block with extended header records. This header block is of the form described in pax Header Block, with a typeflag value of x or g. The extended header records, described in pax Ex? tended Header, shall be included as the data for this header block.
* A header block that describes the file. Any fields in the preceding optional extended header shall override the associated fields in this header block for this file.
* Zero or more blocks that contain the contents of the file.

At the end of the archive file there shall be two 512-byte blocks filled with binary zeros, interpreted as an end-of-archive indicator.

A schematic of an example archive with global extended header records and two actual files is shown in Figure 4-1, pax Format Archive Exam? ple. In the example, the second file in the archive has no extended header preceding it, presumably because it has no need for extended at? tributes.

Figure 4-1: pax Format Archive Example

pax Header Block

The pax header block shall be identical to the ustar header block de?

scribed in ustar Interchange Format, except that two additional type? flag values are defined:

x Represents extended header records for the following file in the archive (which shall have its own ustar header block). The format of these extended header records shall be as described in pax Ex? tended Header.

g Represents global extended header records for the following files in the archive. The format of these extended header records shall be as described in pax Extended Header. Each value shall affect all subsequent files that do not override that value in their own extended header record and until another global extended header record is reached that provides another value for the same field. The typeflag g global headers should not be used with interchange media that could suffer partial data loss in transporting the ar? chive.

For both of these types, the size field shall be the size of the ex? tended header records in octets. The other fields in the header block are not meaningful to this version of the pax utility. However, if this archive is read by a pax utility conforming to the ISO POSIX?2:1993 standard, the header block fields are used to create a regular file that contains the extended header records as data. Therefore, header block field values should be selected to provide reasonable file access to this regular file.

A further difference from the ustar header block is that data blocks for files of typeflag 1 (the digit one) (hard link) may be included, which means that the size field may be greater than zero. Archives cre? ated by pax -o linkdata shall include these data blocks with the hard links.

pax Extended Header

A pax extended header contains values that are inappropriate for the ustar header block because of limitations in that format: fields re? quiring a character encoding other than that described in the ISO/IEC 646:1991 standard, fields representing file attributes not de?

scribed in the ustar header, and fields whose format or length do not fit the requirements of the ustar header. The values in an extended header add attributes to the following file (or files; see the descrip? tion of the typeflag g header block) or override values in the follow? ing header block(s), as indicated in the following list of keywords.

An extended header shall consist of one or more records, each construc? ted as follows:

"%d %s=%s\n", <length>, <keyword>, <value>

The extended header records shall be encoded according to the ISO/IEC 10646?1:2000 standard UTF?8 encoding. The <length> field, <blank>, <equals-sign>, and <newline> shown shall be limited to the portable character set, as encoded in UTF?8. The <keyword> fields can be any UTF?8 characters. The <length> field shall be the decimal length of the extended header record in octets, including the trailing <newline>. If there is a hdrcharset extended header in effect for a file, the value field for any gname, linkpath, path, and uname extended header records shall be encoded using the character set specified by the hdrcharset extended header record; otherwise, the value field shall be encoded using UTF?8. The value field for all other keywords speci? fied by POSIX.1?2008 shall be encoded using UTF?8.

The <keyword> field shall be one of the entries from the following list or a keyword provided as an implementation extension. Keywords con? sisting entirely of lowercase letters, digits, and periods are reserved for future standardization. A keyword shall not include an <equals-sign>. (In the following list, the notations ``file(s)'' or ``block(s)'' is used to acknowledge that a keyword affects the follow? ing single file after a typeflag x extended header, but possibly multi? ple files after typeflag g. Any requirements in the list for pax to include a record when in write or copy mode shall apply only when such a record has not already been provided through the use of the -o op? tion. When used in copy mode, pax shall behave as if an archive had been created with applicable extended header records and then ex? tracted.)

atime The file access time for the following file(s), equivalent to the value of the st_atime member of the stat structure for a file, as described by the stat() function. The access time shall be restored if the process has appropriate privileges required to do so. The format of the <value> shall be as de? scribed in pax Extended Header File Times.

charset The name of the character set used to encode the data in the following file(s). The entries in the following table are de? fined to refer to known standards; additional names may be agreed on between the originator and recipient.

<value>	Formal Standard
ISO-IR 646 1990	ISO/IEC 646:1990
ISO-IR 8859 1 1998	ISO/IEC 8859?1:1998
ISO-IR 8859 2 1999	ISO/IEC 8859?2:1999
ISO-IR 8859 3 1999	ISO/IEC 8859?3:1999
ISO-IR 8859 4 1998	ISO/IEC 8859?4:1998
ISO-IR 8859 5 1999	ISO/IEC 8859?5:1999
ISO-IR 8859 6 1999	ISO/IEC 8859?6:1999
ISO-IR 8859 7 1987	ISO/IEC 8859?7:1987
ISO-IR 8859 8 1999	ISO/IEC 8859?8:1999
ISO-IR 8859 9 1999	ISO/IEC 8859?9:1999
ISO-IR 8859 10 1998	ISO/IEC 8859?10:1998
ISO-IR 8859 13 1998	ISO/IEC 8859?13:1998
ISO-IR 8859 14 1998	ISO/IEC 8859?14:1998
ISO-IR 8859 15 1999	ISO/IEC 8859?15:1999
ISO-IR 10646 2000	ISO/IEC 10646:2000
ISO-IR 10646 2000 UTF-8	ISO/IEC 10646, UTF-8 encoding
BINARY	None.

The encoding is included in an extended header for informa? tion only; when pax is used as described in POSIX.1?2008, it

shall not translate the file data into any other encoding.

The BINARY entry indicates unencoded binary data.

When used in write or copy mode, it is implementation-defined whether pax includes a charset extended header record for a file.

comment A series of characters used as a comment. All characters in the <value> field shall be ignored by pax.

gid The group ID of the group that owns the file, expressed as a decimal number using digits from the ISO/IEC 646:1991 stan? dard. This record shall override the gid field in the follow? ing header block(s). When used in write or copy mode, pax shall include a gid extended header record for each file whose group ID is greater than 2097151 (octal 7777777).

gname The group of the file(s), formatted as a group name in the group database. This record shall override the gid and gname fields in the following header block(s), and any gid extended header record. When used in read, copy, or list mode, pax shall translate the name from the encoding in the header record to the character set appropriate for the group data? base on the receiving system. If any of the characters cannot be translated, and if neither the -oinvalid=UTF?8 option nor the -oinvalid=binary option is specified, the results are im? plementation-defined. When used in write or copy mode, pax shall include a gname extended header record for each file whose group name cannot be represented entirely with the let? ters and digits of the portable character set.

hdrcharset

The name of the character set used to encode the value field of the gname, linkpath, path, and uname pax extended header records. The entries in the following table are defined to refer to known standards; additional names may be agreed be? tween the originator and the recipient.

? <value> ? Formal Standard ?

??

?ISO-IR 10646 2000 UTF-8 ? ISO/IEC 10646, UTF-8 encoding ?

?BINARY ? None. ?

??

If no hdrcharset extended header record is specified, the de? fault character set used to encode all values in extended header records shall be the ISO/IEC 10646?1:2000 standard UTF?8 encoding.

The BINARY entry indicates that all values recorded in ex? tended headers for affected files are unencoded binary data from the underlying system.

linkpath The pathname of a link being created to another file, of any type, previously archived. This record shall override the linkname field in the following ustar header block(s). The following ustar header block shall determine the type of link created. If typeflag of the following header block is 1, it shall be a hard link. If typeflag is 2, it shall be a sym? bolic link and the linkpath value shall be the contents of the symbolic link. The pax utility shall translate the name of the link (contents of the symbolic link) from the encoding in the header to the character set appropriate for the local file system. When used in write or copy mode, pax shall in? clude a linkpath extended header record for each link whose pathname cannot be represented entirely with the members of the portable character set other than NUL.

mtime The file modification time of the following file(s), equiva? lent to the value of the st_mtime member of the stat struc? ture for a file, as described in the stat() function. This record shall override the mtime field in the following header block(s). The modification time shall be restored if the process has appropriate privileges required to do so. The format of the <value> shall be as described in pax Extended

Header File Times.

path The pathname of the following file(s). This record shall override the name and prefix fields in the following header block(s). The pax utility shall translate the pathname of the file from the encoding in the header to the character set ap? propriate for the local file system.

When used in write or copy mode, pax shall include a path ex? tended header record for each file whose pathname cannot be represented entirely with the members of the portable charac? ter set other than NUL.

realtime.any

The keywords prefixed by ``realtime.'' are reserved for fu? ture standardization.

security.any

The keywords prefixed by ``security.'' are reserved for fu? ture standardization.

size The size of the file in octets, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the size field in the following header block(s). When used in write or copy mode, pax shall include a size extended header record for each file with a size value greater than 8589934591 (octal 77777777777).

uid The user ID of the file owner, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the uid field in the following header block(s). When used in write or copy mode, pax shall include a uid extended header record for each file whose owner ID is greater than 2097151 (octal 7777777).

uname The owner of the following file(s), formatted as a user name in the user database. This record shall override the uid and uname fields in the following header block(s), and any uid extended header record. When used in read, copy, or list mode, pax shall translate the name from the encoding in the

header record to the character set appropriate for the user database on the receiving system. If any of the characters cannot be translated, and if neither the -oinvalid=UTF?8 option nor the -oinvalid=binary option is specified, the results are implementation-defined. When used in write or copy mode, pax shall include a uname extended header record for each file whose user name cannot be represented entirely with the letters and digits of the portable character set.

If the <value> field is zero length, it shall delete any header block field, previously entered extended header value, or global extended header value of the same name.

If a keyword in an extended header record (or in a -o option-argument) overrides or deletes a corresponding field in the ustar header block, pax shall ignore the contents of that header block field.

Unlike the ustar header block fields, NULs shall not delimit <value>s; all characters within the <value> field shall be considered data for the field. None of the length limitations of the ustar header block fields in Table 4-14, ustar Header Block shall apply to the extended header records.

pax Extended Header Keyword Precedence

This section describes the precedence in which the various header records and fields and command line options are selected to apply to a file in the archive. When pax is used in read or list modes, it shall determine a file attribute in the following sequence:

1. If -odelete=keyword-prefix is used, the affected attributes shall be determined from step 7., if applicable, or ignored otherwise.
2. If -okeyword:= is used, the affected attributes shall be ignored.
3. If -okeyword:=value is used, the affected attribute shall be assigned the value.
4. If there is a typeflag x extended header record, the affected attribute shall be assigned the <value>. When extended header records conflict, the last one given in the header shall take precedence.
5. If -okeyword=value is used, the affected attribute shall be as?

signed the value.

6. If there is a typeflag g global extended header record, the af? fected attribute shall be assigned the <value>. When global ex? tended header records conflict, the last one given in the global header shall take precedence.

7. Otherwise, the attribute shall be determined from the ustar header block.

pax Extended Header File Times

The pax utility shall write an mtime record for each file in write or copy modes if the file's modification time cannot be represented ex? actly in the ustar header logical record described in ustar Interchange Format. This can occur if the time is out of ustar range, or if the file system of the underlying implementation supports non-integer time granularities and the time is not an integer. All of these time records shall be formatted as a decimal representation of the time in seconds since the Epoch. If a <period> ('.') decimal point character is present, the digits to the right of the point shall represent the units of a subsecond timing granularity, where the first digit is tenths of a second and each subsequent digit is a tenth of the previous digit. In read or copy mode, the pax utility shall truncate the time of a file to the greatest value that is not greater than the input header file time. In write or copy mode, the pax utility shall output a time exactly if it can be represented exactly as a decimal number, and otherwise shall generate only enough digits so that the same time shall be recovered if the file is extracted on a system whose underlying implementation sup? ports the same time granularity.

ustar Interchange Format

A ustar archive tape or file shall contain a series of logical records. Each logical record shall be a fixed-size logical record 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 logical record that describes the file, fol?

lowed by zero or more logical records that give the contents of the file. At the end of the archive file there shall be two 512-octet logi? cal records filled with binary zeros, interpreted as an end-of-archive indicator.

The logical records may be grouped for physical I/O operations, as de? scribed under the -bblocksize and -x ustar options. Each group of logi? cal records may be written with a single operation equivalent to the write() function. On magnetic tape, the result of this write shall be a single tape physical block. The last physical block shall always be the full size, so logical records after the two zero logical records may contain undefined data.

The header logical record shall be structured as shown in the following table. All lengths and offsets are in decimal.

Table 4-14: ustar 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 logical record shall be represented in the coded character set of the ISO/IEC 646:1991 standard. 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 of <slash> and the portable filename charac? ter set in names for files, users, and groups, one or more implementa? tion-defined encodings of these characters shall be provided for inter? change purposes.

However, the pax utility shall never create filenames on the local sys? tem that cannot be accessed via the procedures described in POSIX.1?2008. If a filename is found on the medium that would create an invalid filename, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored. The pax 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 logical record is contiguous; that is, there is no padding used. Each character on the archive medium shall be stored contiguously.

The fields magic, uname, and gname are character strings each termi? nated by a NUL character. The fields name, linkname, and prefix are NUL-terminated character strings except when all characters in the ar? ray contain non-NUL characters including the last character. The ver? sion field is two octets containing the characters "00" (zero-zero).

The typeflag contains a single character. All other fields are leading zero-filled octal numbers using digits from the ISO/IEC 646:1991 stan? dard IRV. Each numeric field is terminated by one or more <space> or NUL characters.

The name and the prefix fields shall produce the pathname of the file. 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 is used

alone. In either case, name is 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, pax shall notify the user of the error, and shall not store any part of the file?header or data? on the medium.

The linkname field, described below, shall not use the prefix to pro? duce a pathname. As such, a linkname is limited to 100 characters. If the name does not fit in the space provided, pax shall notify the user of the error, and shall not attempt to store the link on the medium.

The mode field provides 12 bits encoded in the ISO/IEC 646:1991 stan? dard octal digit representation. The encoded bits shall represent the following values:

Table: ustar mode Field

Bit Value	POSIX.1?2008 Bit	Description
04000	S_ISUID	Set UID on execution.
02000	S_ISGID	Set GID on execution.
01000	<reserved>	Reserved for future standardization.
00400	S_IRUSR	Read permission for file owner class.
00200	S_IWUSR	Write permission for file owner class.
00100	S_IXUSR	Execute/search permission for file owner class.
00040	S_IRGRP	Read permission for file group class.
00020	S_IWGRP	Write permission for file group class.
00010	S_IXGRP	Execute/search permission for file group class.
00004	S_IROTH	Read permission for file other class.
00002	S_IWOTH	Write permission for file other class.
00001	S_IXOTH	Execute/search permission for file other class.

When appropriate privileges are required to set one of these mode bits, and the user restoring the files from the archive does not have appro? priate privileges, the mode bits for which the user does not have ap?

propriate privileges shall be ignored. Some of the mode bits in the archive format are not mentioned elsewhere in this volume of POSIX.1?2017. If the implementation does not support those bits, they may be ignored.

The uid and gid fields are the user and group ID of the owner and group of the file, respectively.

The size field is 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 (a symbolic link), 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 logical records are 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 volume of POSIX.1?2017, and no data logical records shall be stored on the medium. Additionally, for type 6, the size field shall be ignored when reading. If the typeflag field is set to any other value, the number of logical records 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 646:1991 standard representation of the octal value of the modification time obtained from the stat() function.

The chksum field shall be the ISO/IEC 646:1991 standard IRV representation of the octal value of the simple sum of all octets in the header logical record. 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 is not less than 17 bits. When calculating the checksum, the chksum field is treated as if it were all <space> characters.

The typeflag field specifies the type of file archived. If a particular implementation does not recognize the type, or the user does not have appropriate privileges 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 logical records to be written on the medium (see the previous description for size). If con? version to a regular file occurs, the pax utility shall produce an er? ror indicating that the conversion took place. All of the typeflag fields shall be coded in the ISO/IEC 646:1991 standard IRV:

0 Represents a regular file. For backwards-compatibility, a type? flag value of binary zero ('\0') should be recognized as mean? ing a regular file when extracting files from the archive. Ar? chives written with this version of the archive file format create regular files with a typeflag value of the ISO/IEC 646:1991 standard IRV '0'.

1 Represents a file linked to another file, of any type, previ? ously archived. Such files are identified by having the same device and file serial numbers, and pathnames that refer to different directory entries. All such files shall be archived as linked files. The linked-to name is specified in the linkname field with a NUL-character terminator if it is less than 100 octets in length.

2 Represents a symbolic link. The contents of the symbolic link shall be stored in the linkname field.

3,4 Represent character special files and block special files re? spectively. In this case the devmajor and devminor fields shall contain information defining the device, the format of which is unspecified by this volume of POSIX.1?2017. Implemen? tations 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 di? rectory may hold. A size field of zero indicates no such lim? iting. 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 con? tents.

7 Reserved to represent a file to which an implementation has as? sociated some high-performance attribute. Implementations with? out such extensions should treat this file as a regular file (type 0).

A?Z The letters 'A' to 'Z', inclusive, are reserved for custom im? plementations. All other values are reserved for future ver? sions of this standard.

It is unspecified whether files with pathnames that refer to the same directory entry are archived as linked files or as separate files. If they are archived as linked files, this means that attempting to ex? tract both pathnames from the resulting archive will always cause an error (unless the -u option is used) because the link cannot be cre? ated.

It is unspecified whether files with the same device and file serial numbers being appended to an archive are treated as linked files to members that were in the archive before the append.

Attempts to archive a socket shall produce a diagnostic message when ustar interchange format is used, but may be allowed when pax inter? change format is used. Handling of other file types is implementation-defined.

The magic field is the specification that this archive was output in this archive format. If this field contains ustar (the five characters from the ISO/IEC 646:1991 standard IRV shown followed by NUL), the un? ame and gname fields shall contain the ISO/IEC 646:1991 standard IRV representation of the owner and group of the file, respectively (trun? cated to fit, if necessary). When the file is restored by a privileged, protection-preserving version of the utility, the user and group data? bases 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.

cpio Interchange Format

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 shall always 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 the following table; see also the <cpio.h> header.

Table 4-16: 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

Filename Field Name	Length	Interpreted as
c_name	c_namesize	Pathname string

File Data Field Name	Length	Interpreted as

?c_filedata c_filesize Data ?

??

cpio Header

For each file in the archive, a header as defined previously shall be written. The information in the header fields is written as streams of the ISO/IEC 646:1991 standard characters interpreted as octal numbers. The octal numbers shall be extended to the necessary length by append? ing the ISO/IEC 646:1991 standard IRV zeros at the most-significant- digit end of the number; the result is written to the most-significant digit of the stream of octets first. The fields shall be interpreted as follows:

c_magic Identify the archive as being a transportable archive by con? taining the identifying value "070707".

c_dev, c_ino

Contains values that uniquely identify the file within the archive (that is, no files 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 Contains the file type and access permissions as defined in the following table.

Table 4-17: Values for cpio c_mode Field

??

? File Permissions Name? Value ? Indicates ?

??

? C_IRUSR ? 000400? Read by owner ?

? C_IWUSR ? 000200? Write by owner ?

? C_IXUSR ? 000100? Execute by owner ?

? C_IRGRP ? 000040? Read by group ?

? C_IWGRP ? 000020? Write by group ?

? C_IXGRP ? 000010? Execute by group ?

? C_IROTH ? 000004? Read by others ?

? C_IWOTH ? 000002? Write by others ?

? C_IXOTH ? 000001? Execute by others ?

? C_ISUID ? 004000? Set uid ?

? C_ISGID ? 002000? Set gid ?

? C_ISVTX ? 001000? Reserved ?

??

? File Type Name ? Value ? Indicates ?

??

? C_ISDIR ? 040000? Directory ?

? C_ISFIFO ? 010000? FIFO ?

? C_ISREG ? 0100000? Regular file ?

? C_ISLNK ? 0120000? Symbolic link ?

? ? ? ? ?

?C_ISBLK ? 060000 ? Block special file ?

?C_ISCHR ? 020000 ? Character special file ?

?C_ISSOCK ? 0140000 ? Socket ?

? ? ? ? ?

?C_ISCTG ? 0110000 ? Reserved ?

??

Directories, FIFOs, symbolic links, and regular files shall be supported on a system conforming to this volume of POSIX.1?2017; additional values defined previously are re? served for compatibility with existing systems. Additional file types may be supported; however, such files should not be written to archives intended to be transported to other systems.

c_uid Contains the user ID of the owner.

c_gid Contains the group ID of the group.

c_nlink Contains a number greater than or equal to the number of links in the archive referencing the file. If the -a option is used to append to a cpio archive, then the pax utility need not account for the files in the existing part of the archive when calculating the c_nlink values for the appended part of the archive, and need not alter the c_nlink values in the existing part of the archive if additional files with the

same c_dev and c_ino values are appended to the archive.

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

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

c_namesize

Contains the length of the pathname, including the terminat? ing NUL character.

c_filesize

Contains the length in octets of the data section following the header structure.

cpio Filename

The c_name field shall contain the pathname of the file. The length of this field in octets is the value of c_namesize.

If a filename is found on the medium that would create an invalid path? name, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored.

All characters shall be represented in the ISO/IEC 646:1991 standard 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 pax utility shall never create filenames on the local system that cannot be accessed via the procedures described pre? viously in this volume of POSIX.1?2017. If a filename is found on the medium that would create an invalid filename, it is implementation-de? fined whether the data from the file is stored on the local file system and under what name it is stored. The pax utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

cpio File Data

Following c_name, there shall be c_filesize octets of data. Interpreta? tion of such data occurs in a manner dependent on the file. For regular files, the data shall consist of the contents of the file. For symbolic links, the data shall consist of the contents of the symbolic link. If c_filesize is zero, no data shall be contained in c_filedata.

When restoring from an archive:

* If the user does not have appropriate privileges to create a file of the specified type, pax shall ignore the entry and write an er? ror message to standard error.
* Only regular files and symbolic links 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 privileges 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 volume of POSIX.1?2017. If the implementation does not support those flags, they may be ignored.

cpio Special Entries

FIFO special files, directories, and the trailer shall be recorded with c_filesize equal to zero. Symbolic links shall be recorded with c_file? size equal to the length of the contents of the symbolic link. For other special files, c_filesize is unspecified by this volume of POSIX.1?2017. 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 filename 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.

EXIT STATUS

The following exit values shall be returned:

0 All files were processed successfully.

>0 An error occurred.

CONSEQUENCES OF ERRORS

If pax cannot create a file or a link when reading an archive or cannot find a file when writing an archive, or cannot preserve the user ID, group ID, or file mode when the -p option is specified, a diagnostic message shall be written to standard error and a non-zero exit status shall be returned, but processing shall continue. In the case where pax cannot create a link to a file, pax shall not, by default, create a second copy of the file.

If the extraction of a file from an archive is prematurely terminated by a signal or error, pax may have only partially extracted the file or (if the -n option was not specified) may have extracted a file of the same name as that specified by the user, but which is not the file the user wanted. Additionally, the file modes of extracted directories may have additional bits from the S_IRWXU mask set as well as incorrect modification and access times.

The following sections are informative.

APPLICATION USAGE

Caution is advised when using the -a option to append to a cpio format archive. If any of the files being appended happen to be given the same c_dev and c_ino values as a file in the existing part of the archive, then they may be treated as links to that file on extraction. Thus, it is risky to use -a with cpio format except when it is done on the same system that the original archive was created on, and with the same pax utility, and in the knowledge that there has been little or no file system activity since the original archive was created that could lead to any of the files appended being given the same c_dev and c_ino val? ues as an unrelated file in the existing part of the archive. Also, when (intentionally) appending additional links to a file in the exist? ing part of the archive, the c_nlink values in the modified archive can be smaller than the number of links to the file in the archive, which may mean that the links are not preserved on extraction.

The -p (privileges) option was invented to reconcile differences be? tween historical tar and cpio implementations. In particular, the two utilities use -m in diametrically opposed ways. The -p option also pro?

vides a consistent means of extending the ways in which future file at? tributes can be addressed, such as for enhanced security systems or high-performance files. Although it may seem complex, there are really two modes that are most commonly used:

-p e ``Preserve everything''. This would be used by the historical superuser, someone with all appropriate privileges, to preserve all aspects of the files as they are recorded in the archive. The e flag is the sum of o and p, and other implementation-de? fined attributes.

-p p ``Preserve'' the file mode bits. This would be used by the user with regular privileges who wished to preserve aspects of the file other than the ownership. The file times are preserved by default, but two other flags are offered to disable these and use the time of extraction.

The one pathname per line format of standard input precludes pathnames containing <newline> characters. Although such pathnames violate the portable filename guidelines, they may exist and their presence may in? hibit usage of pax within shell scripts. This problem is inherited from historical archive programs. The problem can be avoided by listing filename arguments on the command line instead of on standard input.

It is almost certain that appropriate privileges are required for pax to accomplish parts of this volume of POSIX.1?2017. Specifically, cre? ating files of type block special or character special, restoring file access times unless the files are owned by the user (the -t option), or preserving file owner, group, and mode (the -p option) all probably re? quire appropriate privileges.

In read mode, implementations are permitted to overwrite files when the archive has multiple members with the same name. This may fail if per? missions on the first version of the file do not permit it to be over? written.

The cpio and ustar formats can only support files up to 8589934592 bytes (8 ? 2^30) in size.

When archives containing binary header information are listed , the

filenames printed may cause strange behavior on some terminals.

When all of the following are true:

1. A file of type directory is being placed into an archive.
2. The ustar archive format is being used.
3. The pathname of the directory is less than or equal to 155 bytes long (it will fit in the prefix field in the ustar header block).
4. The last component of the pathname of the directory is longer than 100 bytes long (it will not fit in the name field in the ustar header block).

some implementations of the pax utility will place the entire directory pathname in the prefix field, set the name field to an empty string, and place the directory in the archive. Other implementations of the pax utility will give an error under these conditions because the name field is not large enough to hold the last component of the directory name. This standard allows either behavior. However, when extracting a directory from a ustar format archive, this standard requires that all implementations be able to extract a directory even if the name field contains an empty string as long as the prefix field does not also con? tain an empty string.

EXAMPLES

The following command:

```
pax -w -f /dev/rmt/1m .
```

copies the contents of the current directory to tape drive 1, medium density (assuming historical System V device naming procedures?the his? torical BSD device name would be /dev/rmt9).

The following commands:

```
mkdir newdir
pax -rw olddir newdir
```

copy the olddir directory hierarchy to newdir.

```
pax -r -s ',^//*usr//*,,' -f a.pax
```

reads the archive a.pax, with all files rooted in /usr in the archive extracted relative to the current directory.

Using the option:

-o listopt="%M %(atime)T %(size)D %(name)s"

overrides the default output description in Standard Output and instead writes:

-rw-rw--- Jan 12 15:53 2003 1492 /usr/foo/bar

Using the options:

-o listopt='%L\t%(size)D\n%.7' \

-o listopt='(name)s\n%(atime)T\n%T'

overrides the default output description in Standard Output and instead writes:

/usr/foo/bar -> /tmp 1492

/usr/fo

Jan 12 15:53 1991

Jan 31 15:53 2003

RATIONALE

The pax utility was new for the ISO POSIX?2:1993 standard. It repre? sents a peaceful compromise between advocates of the historical tar and cpio utilities.

A fundamental difference between cpio and tar was in the way directo? ries were treated. The cpio utility did not treat directories differ? ently from other files, and to select a directory and its contents re? quired that each file in the hierarchy be explicitly specified. For tar, a directory matched every file in the file hierarchy it rooted.

The pax utility offers both interfaces; by default, directories map into the file hierarchy they root. The -d option causes pax to skip any file not explicitly referenced, as cpio historically did. The tar -style behavior was chosen as the default because it was believed that this was the more common usage and because tar is the more commonly available interface, as it was historically provided on both System V and BSD implementations.

The data interchange format specification in this volume of POSIX.1?2017 requires that processes with ``appropriate privileges'' shall always restore the ownership and permissions of extracted files exactly as archived. If viewed from the historic equivalence between

superuser and ``appropriate privileges'', there are two problems with this requirement. First, users running as superusers may unknowingly set dangerous permissions on extracted files. Second, it is needlessly limiting, in that superusers cannot extract files and own them as supe? ruser unless the archive was created by the superuser. (It should be noted that restoration of ownerships and permissions for the superuser, by default, is historical practice in cpio, but not in tar.) In order to avoid these two problems, the pax specification has an additional ``privilege'' mechanism, the -p option. Only a pax invocation with the privileges needed, and which has the -p option set using the e specifi? cation character, has appropriate privileges to restore full ownership and permission information.

Note also that this volume of POSIX.1?2017 requires that the file own? ership and access permissions shall be set, on extraction, in the same fashion as the creat() function when provided with the mode stored in the archive. This means that the file creation mask of the user is ap? plied to the file permissions.

Users should note that directories may be created by pax while extract? ing files with permissions that are different from those that existed at the time the archive was created. When extracting sensitive informa? tion into a directory hierarchy that no longer exists, users are en? couraged to set their file creation mask appropriately to protect these files during extraction.

The table of contents output is written to standard output to facili? tate pipeline processing.

An early proposal had hard links displaying for all pathnames. This was removed because it complicates the output of the case where -v is not specified and does not match historical cpio usage. The hard-link in? formation is available in the -v display.

The description of the -l option allows implementations to make hard links to symbolic links. Earlier versions of this standard did not specify any way to create a hard link to a symbolic link, but many im? plementations provided this capability as an extension. If there are

hard links to symbolic links when an archive is created, the implemen? tation is required to archive the hard link in the archive (unless -H or -L is specified). When in read mode and in copy mode, implementa? tions supporting hard links to symbolic links should use them when ap? propriate.

The archive formats inherited from the POSIX.1?1990 standard have cer? tain restrictions that have been brought along from historical usage. For example, there are restrictions on the length of pathnames stored in the archive. When pax is used in copy(-rw) mode (copying directory hierarchies), the ability to use extensions from the -xpax format over? comes these restrictions.

The default blocksize value of 5120 bytes for cpio was selected because it is one of the standard block-size values for cpio, set when the -B option is specified. (The other default block-size value for cpio is 512 bytes, and this was considered to be too small.) The default block value of 10240 bytes for tar was selected because that is the standard block-size value for BSD tar. The maximum block size of 32256 bytes (215-512 bytes) is the largest multiple of 512 bytes that fits into a signed 16-bit tape controller transfer register. There are known limi? tations in some historical systems that would prevent larger blocks from being accepted. Historical values were chosen to improve compati? bility with historical scripts using dd or similar utilities to manipu? late archives. Also, default block sizes for any file type other than character special file has been deleted from this volume of POSIX.1?2017 as unimportant and not likely to affect the structure of the resulting archive.

Implementations are permitted to modify the block-size value based on the archive format or the device to which the archive is being written. This is to provide implementations with the opportunity to take advan? tage of special types of devices, and it should not be used without a great deal of consideration as it almost certainly decreases archive portability.

The intended use of the -n option was to permit extraction of one or

more files from the archive without processing the entire archive. This was viewed by the standard developers as offering significant perfor? mance advantages over historical implementations. The -n option in early proposals had three effects; the first was to cause special char? acters in patterns to not be treated specially. The second was to cause only the first file that matched a pattern to be extracted. The third was to cause pax to write a diagnostic message to standard error when no file was found matching a specified pattern. Only the second behav? ior is retained by this volume of POSIX.1?2017, for many reasons. First, it is in general not acceptable for a single option to have mul? tiple effects. Second, the ability to make pattern matching characters act as normal characters is useful for parts of pax other than file ex? traction. Third, a finer degree of control over the special characters is useful because users may wish to normalize only a single special character in a single filename. Fourth, given a more general escape mechanism, the previous behavior of the -n option can be easily ob? tained using the -s option or a sed script. Finally, writing a diagnos? tic message when a pattern specified by the user is unmatched by any file is useful behavior in all cases.

In this version, the -n was removed from the copy mode synopsis of pax; it is inapplicable because there are no pattern operands specified in this mode.

There is another method than pax for copying subtrees in POSIX.1?2008 described as part of the cp utility. Both methods are historical prac? tice: cp provides a simpler, more intuitive interface, while pax offers a finer granularity of control. Each provides additional functionality to the other; in particular, pax maintains the hard-link structure of the hierarchy while cp does not. It is the intention of the standard developers that the results be similar (using appropriate option combi? nations in both utilities). The results are not required to be identi? cal; there seemed insufficient gain to applications to balance the dif? ficulty of implementations having to guarantee that the results would be exactly identical.

A single archive may span more than one file. It is suggested that im?
plementations provide informative messages to the user on standard er?
ror whenever the archive file is changed.

The -d option (do not create intermediate directories not listed in the archive) found in early proposals was originally provided as a comple?
ment to the historic -d option of cpio. It has been deleted.

The -s option in early proposals specified a subset of the substitution command from the ed utility. As there was no reason for only a subset to be supported, the -s option is now compatible with the current ed specification. Since the delimiter can be any non-null character, the following usage with single <space> characters is valid:

```
pax -s " foo bar " ...
```

The -t description is worded so as to note that this may cause the ac?
cess time update caused by some other activity (which occurs while the file is being read) to be overwritten.

The default behavior of pax with regard to file modification times is the same as historical implementations of tar. It is not the histori?
cal behavior of cpio.

Because the -i option uses /dev/tty, utilities without a controlling terminal are not able to use this option.

The -y option, found in early proposals, has been deleted because a line containing a single <period> for the -i option has equivalent functionality. The special lines for the -i option (a single <period> and the empty line) are historical practice in cpio.

In early drafts, a -echarmap option was included to increase portabil?
ity of files between systems using different coded character sets. This option was omitted because it was apparent that consensus could not be formed for it. In this version, the use of UTF?8 should be an adequate substitute.

The ISO POSIX?2:1993 standard and ISO POSIX?1 standard requirements for pax, however, made it very difficult to create a single archive con?
taining files created using extended characters provided by different locales. This version adds the hdrcharset keyword to make it possible

to archive files in these cases without dropping files due to transla? tion errors.

Translating filenames and other attributes from a locale's encoding to UTF?8 and then back again can lose information, as the resulting file? name might not be byte-for-byte equivalent to the original. To avoid this problem, users can specify the -o hdrcharset=binary option, which will cause the resulting archive to use binary format for all names and attributes. Such archives are not portable among hosts that use differ? ent native encodings (e.g., EBCDIC versus ASCII-based encodings), but they will allow interchange among the vast majority of POSIX file sys? tems in practical use. Also, the -o hdrcharset=binary option will cause pax in copy mode to behave more like other standard utilities such as cp.

If the values specified by the -o exthdr.name=value, -o globex? thdr.name=value, or by $TMPDIR (if -o globexthdr.name is not specified) require a character encoding other than that described in the ISO/IEC 646:1991 standard, a path extended header record will have to be created for the file. If a hdrcharset extended header record is ac? tive for such headers, it will determine the codeset used for the value field in these extended path header records. These path extended header records always need to be created when writing an archive even if hdr? charset=binary has been specified and would contain the same (binary) data that appears in the ustar header record prefix and name fields. (In other words, an extended header path record is always required to be generated if the prefix or name fields contain non-ASCII characters even when hdrcharset=binary is also in effect for that file.)

The -k option was added to address international concerns about the dangers involved in the character set transformations of -e (if the target character set were different from the source, the filenames might be transformed into names matching existing files) and also was made more general to protect files transferred between file systems with different {NAME_MAX} values (truncating a filename on a smaller system might also inadvertently overwrite existing files). As stated,

it prevents any overwriting, even if the target file is older than the source. This version adds more granularity of options to solve this problem by introducing the -oinvalid=option?specifically the UTF?8 and binary actions. (Note that an existing file is still subject to over? writing in this case. The -k option closes that loophole.)

Some of the file characteristics referenced in this volume of POSIX.1?2017 might not be supported by some archive formats. For exam? ple, neither the tar nor cpio formats contain the file access time. For this reason, the e specification character has been provided, intended to cause all file characteristics specified in the archive to be re? tained.

It is required that extracted directories, by default, have their ac? cess and modification times and permissions set to the values specified in the archive. This has obvious problems in that the directories are almost certainly modified after being extracted and that directory per? missions may not permit file creation. One possible solution is to cre? ate directories with the mode specified in the archive, as modified by the umask of the user, with sufficient permissions to allow file cre? ation. After all files have been extracted, pax would then reset the access and modification times and permissions as necessary.

The list-mode formatting description borrows heavily from the one de? fined by the printf utility. However, since there is no separate oper? and list to get conversion arguments, the format was extended to allow specifying the name of the conversion argument as part of the conver? sion specification.

The T conversion specifier allows time fields to be displayed in any of the date formats. Unlike the ls utility, pax does not adjust the format when the date is less than six months in the past. This makes parsing the output more predictable.

The D conversion specifier handles the ability to display the major/mi? nor or file size, as with ls, by using %-8(size)D.

The L conversion specifier handles the ls display for symbolic links.

Conversion specifiers were added to generate existing known types used

for ls.

pax Interchange Format

The new POSIX data interchange format was developed primarily to satisfy international concerns that the ustar and cpio formats did not provide for file, user, and group names encoded in characters outside a subset of the ISO/IEC 646:1991 standard. The standard developers realized that this new POSIX data interchange format should be very extensible because there were other requirements they foresaw in the near future:

* Support international character encodings and locale information
* Support security information (ACLs, and so on)
* Support future file types, such as realtime or contiguous files
* Include data areas for implementation use
* Support systems with words larger than 32 bits and timers with subsecond granularity

The following were not goals for this format because these are better handled by separate utilities or are inappropriate for a portable format:

* Encryption
* Compression
* Data translation between locales and codesets
* inode storage

The format chosen to support the goals is an extension of the ustar format. Of the two formats previously available, only the ustar format was selected for extensions because:

* It was easier to extend in an upwards-compatible way. It offered version flags and header block type fields with room for future standardization. The cpio format, while possessing a more flexible file naming methodology, could not be extended without breaking some theoretical implementation or using a dummy filename that could be a legitimate filename.
* Industry experience since the original ``tar wars'' fought in developing the ISO POSIX?1 standard has clearly been in favor of the

ustar format, which is generally the default output format selected for pax implementations on new systems.

The new format was designed with one additional goal in mind: reasonable behavior when an older tar or pax utility happened to read an archive. Since the POSIX.1?1990 standard mandated that a ``format-reading utility'' had to treat unrecognized typeflag values as regular files, this allowed the format to include all the extended information in a pseudo-regular file that preceded each real file. An option is given that allows the archive creator to set up reasonable names for these files on the older systems. Also, the normative text suggests that reasonable file access values be used for this ustar header block. Making these header files inaccessible for convenient reading and deleting would not be reasonable. File permissions of 600 or 700 are suggested.

The ustar typeflag field was used to accommodate the additional functionality of the new format rather than magic or version because the POSIX.1?1990 standard (and, by reference, the previous version of pax), mandated the behavior of the format-reading utility when it encountered an unknown typeflag, but was silent about the other two fields.

Early proposals for the first version of this standard contained a proposed archive format that was based on compatibility with the standard for tape files (ISO 1001, similar to the format used historically on many mainframes and minicomputers). This format was overly complex and required considerable overhead in volume and header records. Furthermore, the standard developers felt that it would not be acceptable to the community of POSIX developers, so it was later changed to be a format more closely related to historical practice on POSIX systems.

The prefix and name split of pathnames in ustar was replaced by the single path extended header record for simplicity.

The concept of a global extended header (typeflagg) was controversial. If this were applied to an archive being recorded on magnetic tape, a few unreadable blocks at the beginning of the tape could be a serious problem; a utility attempting to extract as many files as possible from a damaged archive could lose a large percentage of file header informa?

tion in this case. However, if the archive were on a reliable medium, such as a CD?ROM, the global extended header offers considerable poten? tial size reductions by eliminating redundant information. Thus, the text warns against using the global method for unreliable media and provides a method for implanting global information in the extended header for each file, rather than in the typeflag g records.

No facility for data translation or filtering on a per-file basis is included because the standard developers could not invent an interface that would allow this in an efficient manner. If a filter, such as en? cryption or compression, is to be applied to all the files, it is more efficient to apply the filter to the entire archive as a single file.

The standard developers considered interfaces that would invoke a shell script for each file going into or out of the archive, but the system overhead in this approach was considered to be too high.

One such approach would be to have filter= records that give a pathname for an executable. When the program is invoked, the file and archive would be open for standard input/output and all the header fields would be available as environment variables or command-line arguments. The standard developers did discuss such schemes, but they were omitted from POSIX.1?2008 due to concerns about excessive overhead. Also, the program itself would need to be in the archive if it were to be used portably.

There is currently no portable means of identifying the character set(s) used for a file in the file system. Therefore, pax has not been given a mechanism to generate charset records automatically. The only portable means of doing this is for the user to write the archive using the -ocharset=string command line option. This assumes that all of the files in the archive use the same encoding. The ``implementation-de? fined'' text is included to allow for a system that can identify the encodings used for each of its files.

The table of standards that accompanies the charset record description is acknowledged to be very limited. Only a limited number of character set standards is reasonable for maximal interchange. Any character set

is, of course, possible by prior agreement. It was suggested that EBCDIC be listed, but it was omitted because it is not defined by a formal standard. Formal standards, and then only those with reasonably large followings, can be included here, simply as a matter of practi? cality. The <value>s represent names of officially registered character sets in the format required by the ISO 2375:1985 standard.

The normal <comma> or <blank>-separated list rules are not followed in the case of keyword options to allow ease of argument parsing for getopts.

Further information on character encodings is in pax Archive Character Set Encoding/Decoding.

The standard developers have reserved keyword name space for vendor ex? tensions. It is suggested that the format to be used is:

VENDOR.keyword

where VENDOR is the name of the vendor or organization in all uppercase letters. It is further suggested that the keyword following the <pe? riod> be named differently than any of the standard keywords so that it could be used for future standardization, if appropriate, by omitting the VENDOR prefix.

The <length> field in the extended header record was included to make it simpler to step through the records, even if a record contains an unknown format (to a particular pax) with complex interactions of spe? cial characters. It also provides a minor integrity checkpoint within the records to aid a program attempting to recover files from a damaged archive.

There are no extended header versions of the devmajor and devminor fields because the unspecified format ustar header field should be suf? ficient. If they are not, vendor-specific extended keywords (such as VENDOR.devmajor) should be used.

Device and i-number labeling of files was not adopted from cpio; files are interchanged strictly on a symbolic name basis, as in ustar.

Just as with the ustar format descriptions, the new format makes no special arrangements for multi-volume archives. Each of the pax archive

types is assumed to be inside a single POSIX file and splitting that file over multiple volumes (diskettes, tape cartridges, and so on), processing their labels, and mounting each in the proper sequence are considered to be implementation details that cannot be described portably.

The pax format is intended for interchange, not only for backup on a single (family of) systems. It is not as densely packed as might be possible for backup:

* It contains information as coded characters that could be coded in binary.
* It identifies extended records with name fields that could be omit? ted in favor of a fixed-field layout.
* It translates names into a portable character set and identifies locale-related information, both of which are probably unnecessary for backup.

The requirements on restoring from an archive are slightly different from the historical wording, allowing for non-monolithic 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 POSIX.1?2008 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.

Links are recorded in the fashion described here because a link can be to any file type. It is desirable in general to be able to restore part of an archive selectively and restore all of those 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 link name of any link, and the user may choose whether data is recorded with each instance of a

file that contains data. 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 interpret such archives on input properly, despite the fact that there is no pax option that would force this mixed case on output. (When -o linkdata is used, the output must contain the dupli? cate data, but the implementation is free to include it or omit it when -o linkdata is not used.)

The time values are included as extended header records for those im? plementations needing more than the eleven octal digits allowed by the ustar format. Portable file timestamps cannot be negative. If pax en? counters a file with a negative timestamp in copy or write mode, it can reject the file, substitute a non-negative timestamp, or generate a non-portable timestamp with a leading '-'. Even though some implemen? tations can support finer file-time granularities than seconds, the normative text requires support only for seconds since the Epoch be? cause the ISO POSIX?1 standard states them that way. The ustar format includes only mtime; the new format adds atime and ctime for symmetry. The atime access time restored to the file system will be affected by the -p a and -p e options. The ctime creation time (actually inode mod? ification time) is described with appropriate privileges so that it can be ignored when writing to the file system. POSIX does not provide a portable means to change file creation time. Nothing is intended to prevent a non-portable implementation of pax from restoring the value. The gid, size, and uid extended header records were included to allow expansion beyond the sizes specified in the regular tar header. New file system architectures are emerging that will exhaust the 12-digit size field. There are probably not many systems requiring more than 8 digits for user and group IDs, but the extended header values were in? cluded for completeness, allowing overrides for all of the decimal val? ues in the tar header.

The standard developers intended to describe the effective results of pax with regard to file ownerships and permissions; implementations are not restricted in timing or sequencing the restoration of such, pro?

vided the results are as specified.

Much of the text describing the extended headers refers to use in ``write or copy modes''. The copy mode references are due to the norma? tive text: ``The effect of the copy shall be as if the copied files were written to an archive file and then subsequently extracted ...''.

There is certainly no way to test whether pax is actually generating the extended headers in copy mode, but the effects must be as if it had.

pax Archive Character Set Encoding/Decoding

There is a need to exchange archives of files between systems of dif? ferent native codesets. Filenames, group names, and user names must be preserved to the fullest extent possible when an archive is read on the receiving platform. Translation of the contents of files is not within the scope of the pax utility.

There will also be the need to represent characters that are not avail? able on the receiving platform. These unsupported characters cannot be automatically folded to the local set of characters due to the chance of collisions. This could result in overwriting previous extracted files from the archive or pre-existing files on the system.

For these reasons, the codeset used to represent characters within the extended header records of the pax archive must be sufficiently rich to handle all commonly used character sets. The fields requiring transla? tion include, at a minimum, filenames, user names, group names, and link pathnames. Implementations may wish to have localized extended keywords that use non-portable characters.

The standard developers considered the following options:

* The archive creator specifies the well-defined name of the source codeset. The receiver must then recognize the codeset name and per? form the appropriate translations to the destination codeset.
* The archive creator includes within the archive the character map? ping table for the source codeset used to encode extended header records. The receiver must then read the character mapping table and perform the appropriate translations to the destination code?

set.

* The archive creator translates the extended header records in the source codeset into a canonical form. The receiver must then per? form the appropriate translations to the destination codeset.

The approach that incorporates the name of the source codeset poses the problem of codeset name registration, and makes the archive useless to pax archive decoders that do not recognize that codeset.

Because parts of an archive may be corrupted, the standard developers felt that including the character map of the source codeset was too fragile. The loss of this one key component could result in making the entire archive useless. (The difference between this and the global ex? tended header decision was that the latter has a workaround?duplicating extended header records on unreliable media?but this would be too bur? densome for large character set maps.)

Both of the above approaches also put an undue burden on the pax ar? chive receiver to handle the cross-product of all source and destina? tion codesets.

To simplify the translation from the source codeset to the canonical form and from the canonical form to the destination codeset, the stan? dard developers decided that the internal representation should be a stateless encoding. A stateless encoding is one where each codepoint has the same meaning, without regard to the decoder being in a specific state. An example of a stateful encoding would be the Japanese Shift-JIS; an example of a stateless encoding would be the ISO/IEC 646:1991 standard (equivalent to 7-bit ASCII).

For these reasons, the standard developers decided to adopt a canonical format for the representation of file information strings. The obvious, well-endorsed candidate is the ISO/IEC 10646?1:2000 standard (based in part on Unicode), which can be used to represent the characters of vir? tually all standardized character sets. The standard developers ini? tially agreed upon using UCS2 (16-bit Unicode) as the internal repre? sentation. This repertoire of characters provides a sufficiently rich set to represent all commonly-used codesets.

However, the standard developers found that the 16-bit Unicode repre?
sentation had some problems. It forced the issue of standardizing byte
ordering. The 2-byte length of each character made the extended header
records twice as long for the case of strings coded entirely from his?
torical 7-bit ASCII. For these reasons, the standard developers chose
the UTF?8 defined in the ISO/IEC 10646?1:2000 standard. This multi-byte
representation encodes UCS2 or UCS4 characters reliably and determinis?
tically, eliminating the need for a canonical byte ordering. In addi?
tion, NUL octets and other characters possibly confusing to POSIX file
systems do not appear, except to represent themselves. It was realized
that certain national codesets take up more space after the encoding,
due to their placement within the UCS range; it was felt that the use?
fulness of the encoding of the names outweighs the disadvantage of size
increase for file, user, and group names.

The encoding of UTF?8 is as follows:

UCS4 Hex Encoding	UTF-8 Binary Encoding
00000000-0000007F	0xxxxxxx
00000080-000007FF	110xxxxx 10xxxxxx
00000800-0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
00010000-001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
00200000-03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
04000000-7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

where each 'x' represents a bit value from the character being trans?
lated.

ustar Interchange Format

The description of the ustar format reflects numerous enhancements over
pre-1988 versions of the historical tar utility. 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 ar?
chive format are compatible with the new format.

Implementors should be aware that the previous file format did not in?
clude a mechanism to archive directory type files. For this reason, the

convention of using a filename ending with <slash> was adopted to spec?
ify 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 backwards-compatibil?
ity, and instead the prefix was introduced. Also, because of the ear?
lier 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 ex?
tensions, 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 ex?
tension 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 or?
dinary data and conflicts with an existing filename, need not be spe?
cially detected by the utility. If run as an ordinary user, pax 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 simi?
larly named special files; it is impossible to anticipate the needs of the user (who could really intend to overwrite the file), so the behav?
ior should be predictable (and thus regular) and rely on the protection system as required.

The value 7 in the typeflag field is intended to define how contiguous files can be stored in a ustar archive. POSIX.1?2008 does not require

the contiguous file extension, but does define a standard way of ar?
chiving such 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 pax utility should do the best it
can with the file and go on to the next.

The file protection modes are those conventionally used by the ls util?
ity. This is extended beyond the usage in the ISO POSIX?2 standard 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.

cpio Interchange Format

The reference to appropriate privileges in the cpio format refers to an
error on standard output; the ustar format does not make comparable
statements.

The model for this format was the historical System V cpio-c data in?
terchange format. This model documents the portable version of the cpio
format and not the binary version. It has the flexibility to transfer
data of any type described within POSIX.1?2008, yet is extensible to
transfer data types specific to extensions beyond POSIX.1?2008 (for ex?
ample, contiguous files). Because it describes existing practice, there
is no question of maintaining upwards-compatibility.

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 inode
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 inode number of the file in the location from which
that file is extracted.

The name c_magic is based on historical usage.

cpio Filename

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. However, an implementation may use c_namesize to deter? mine the exact length of the pathname. With the current description of the <cpio.h> header, this pathname size can be as large as a number that is described in six octal digits.

Two 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.

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 pax utility should do the best it can with the file and go on to the next.

FUTURE DIRECTIONS

None.

SEE ALSO

Chapter 2, Shell Command Language, cp, ed, getopts, ls, printf

The Base Definitions volume of POSIX.1?2017, Section 3.169, File Mode Bits, Chapter 5, File Format Notation, Chapter 8, Environment Vari? ables, Section 12.2, Utility Syntax Guidelines, <cpio.h>, <tar.h>

The System Interfaces volume of POSIX.1?2017, chown(), creat(), fs? tatat(), mkdir(), mkfifo(), utime(), write()

COPYRIGHT

event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .