pax - the POSIX archiver

pax can do a lot of fancy stuff, feel free to contribute more awesome pax tricks!

Introduction

The POSIX archiver, `pax` , is an attempt at a standardized archiver with the best features of `tar` and `cpio` , able to handle all common archive types.

However, this is **not a manpage**, it will **not** list all possible options, it will **not** you detailed information about `pax` . It's only an introduction.

This article is based on the debianized Berkeley implementation of `pax` , but implementation-specific things should be tagged as such. Unfortunately, the Debian package doesn't seem to be maintained anymore.

Overview

Operation modes

There are four basic operation modes to *list*, *read*, *write* and *copy* archives. They're switched with combinations of `-r` and `-w` command line options:

Mode	RW-Options
List	*no RW-options*
Read	`-r`
Write	`-w`
Copy	`-r -w`

List

In *list mode*, `pax` writes the list of archive members to standard output (a table of contents). If a pattern match is specified on the command line, only matching filenames are printed.

Read

Read an archive. `pax` will read archive data and extract the members to the current directory. If a pattern match is specified on the command line, only matching filenames are extracted.

When reading an archive, the archive type is determined from the archive data.

Write

Write an archive, which means create a new one or append to an existing one. All files and directories specified on the command line are inserted into the archive. The archive is written to standard output by default.

If no files are specified on the command line, filenames are read from `STDIN` .

The write mode is the only mode where you need to specify the archive type with `-x <TYPE>` , e.g. `-x ustar` .

Copy

Copy mode is similar to `cpio` passthrough mode. It provides a way to replicate a complete or partial file hierarchy (with all the `pax` options, e.g. rewriting groups) to another location.

Archive data

When you don't specify anything special, `pax` will attempt to read archive data from standard input (read/list modes) and write archive data to standard output (write mode). This ensures `pax` can be easily used as part of a shell pipe construct, e.g. to read a compressed archive that's decompressed in the pipe.

The option to specify the pathname of a file to be archived is `-f` This file will be used as input or output, depending on the operation (read/write/list).

When pax reads an archive, it tries to guess the archive type. However, in *write* mode, you must specify which type of archive to append using the `-x <TYPE>` switch. If you omit this switch, a default archive will be created (POSIX says it's implementation defined, Berkeley `pax` creates `ustar` if no options are specified).

The following archive formats are supported (Berkeley implementation):

ustar	POSIX TAR format (default)
cpio	POSIX CPIO format
tar	classic BSD TAR format
bcpio	old binary CPIO format
sv4cpio	SVR4 CPIO format
sv4crc	SVR4 CPIO format with CRC

Berkeley `pax` supports options `-z` and `-j` , similar to GNU `tar` , to filter archive files through GZIP/BZIP2.

Matching archive members

In *read* and *list* modes, you can specify patterns to determine which files to list or extract.

- the pattern notation is the one known by a POSIX-shell, i.e. the one known by Bash without `extglob`
- if the specified pattern matches a complete directory, it affects all files and subdirectories of the specified directory
- if you specify the `-c` option, `pax` will invert the matches, i.e. it matches all filenames **except** those matching the specified patterns
- if no patterns are given, `pax` will "match" (list or extract) all files from the archive
- **To avoid conflicts with shell pathname expansion, it's wise to quote patterns!**

Some assorted examples of patterns

```
pax -r <myarchive.tar 'data/sales/*.txt' 'data/products/*.png'
```

```
pax -r <myarchive.tar 'data/sales/year_200[135].txt'
# should be equivalent to
pax -r <myarchive.tar 'data/sales/year_2001.txt' 'data/sales/year_200
3.txt' 'data/sales/year_2005.txt'
```

Using pax

This is a brief description of using `pax` as a normal archiver system, like you would use `tar`.

Creating an archive

This task is done with basic syntax

```
# archive contents to stdout
pax -w >archive.tar README.txt *.png data/

# equivalent, extract archive contents directly to a file
pax -w -x ustar -f archive.tar README.txt *.png data/
```

`pax` is in *write* mode, the given filenames are packed into an archive:

- `README.txt` is a normal file, it will be packed
- `*.png` is a pathname glob **for your shell**, the shell will substitute all matching filenames **before** `pax` is executed. The result is a list of filenames that will be packed like the `README.txt` example above
- `data/` is a directory. **Everything** in this directory will be packed into the archive, i.e. not just an empty directory

When you specify the `-v` option, `pax` will write the pathnames of the files inserted into the archive to `STDERR`.

When, and only when, no filename arguments are specified, `pax` attempts to read filenames from `STDIN`, separated by newlines. This way you can easily combine `find` with `pax`:

```
find . -name '*.txt' | pax -wf textfiles.tar -x ustar
```

Listing archive contents

The standard output format to list archive members simply is to print each filename to a separate line. But the output format can be customized to include permissions, timestamps, etc. with the `-o listopt=<FORMAT>` specification. The syntax of the format specification is strongly derived from the `printf(3)` format specification.

Unfortunately the `pax` utility delivered with Debian doesn't seem to support these extended listing formats.

However, `pax` lists archive members in a `ls -l`-like format, when you give the `-v` option:

```
pax -v <myarchive.tar
# or, of course
pax -vf myarchive.tar
```

Extracting from an archive

You can extract all files, or files (not) matching specific patterns from an archive using constructs like:

```
# "normal" extraction
pax -rf myarchive.tar '*.txt'
```

```
# with inverted pattern
pax -rf myarchive.tar -c '*.txt'
```

Copying files

To copy directory contents to another directory, similar to a `cp -a` command, use:

```
mkdir destdir
pax -rw dir destdir #creates a copy of dir in destdir/, i.e. destdir/
dir
```

Copying files via ssh

To copy directory contents to another directory on a remote system, use:

```
pax -w localdir | ssh user@host "cd distantdest && pax -r -v"
pax -w localdir | gzip | ssh user@host "cd distantdir && gunzip | pax
-r -v" #compress the sent data
```

These commands create a copy of localdir in distandir (distantdir/dir) on the remote machine.

Advanced usage

Backup your daily work

Note: `-T` is an extension and is not defined by POSIX.

Say you have write-access to a fileserver mounted on your filesystem tree. In *copy* mode, you can tell `pax` to copy only files that were modified today:

```
mkdir /n/mybackups/$(date +%A)/
pax -rw -T 0000 data/ /n/mybackups/$(date +%A)/
```

This is done using the `-T` switch, which normally allows you to specify a time window, but in this case, only the start time which means "today at midnight".

When you execute this "very simple backup" after your daily work, you will have a copy of the modified files.

Note: The `%A` format from `date` expands to the name of the current day, localized, e.g. "Friday" (en) or "Mittwoch" (de).

The same, but with an archive, can be accomplished by:

```
pax -w -T 0000 -f /n/mybackups/$(date +%A)
```

In this case, the day-name is an archive-file (you don't need a filename extension like `.tar` but you can add one, if desired).

Changing filenames while archiving

`pax` is able to rewrite filenames while archiving or while extracting from an archive. This example creates a tar archive containing the `holiday_2007/` directory, but the directory name inside the archive will be `holiday_pics/`:

```
pax -x ustar -w -f holiday_pictures.tar -s '/^holiday_2007/holiday_pi
cs/' holiday_2007/
```

The option responsible for the string manipulation is the `-s <REWRITE-SPECIFICATION>`. It takes the string rewrite specification as an argument, in the form `/OLD/NEW/[gp]`, which is an `ed(1)` -like regular expression (BRE) for `old` and generally can be used like the popular sed construct `s/from/to/`. Any non-null character can be used as a delimiter, so to mangle pathnames (containing slashes), you could use `#/old/path#/new/path#`.

The optional `g` and `p` flags are used to apply substitution **(g)**lobally to the line or to **(p)**rint the original and rewritten strings to `STDERR`.

Multiple `-s` options can be specified on the command line. They are applied to the pathname strings of the files or archive members. This happens in the order they are specified.

Excluding files from an archive

The -s command seen above can be used to exclude a file. The substitution must result in a null string: For example, let's say that you want to exclude all the CVS directories to create a source code archive. We are going to replace the names containing /CVS/ with nothing, note the .* they are needed because we need to match the entire pathname.

```
  pax -w -x ustar -f release.tar -s',.*/CVS/.*,,' myapplication
```

You can use several -s options, for instance, let's say you also want to remove files ending in ~:

```
  pax -w -x ustar -f release.tar -'s,.*/CVS/.*,,' -'s/.*~//' myapplic
ation
```

This can also be done while reading an archive, for instance, suppose you have an archive containing a "usr" and a "etc" directory but that you want to extract only the "usr" directory:

```
pax -r -f archive.tar -s',^etc/.*,,' #the etc/ dir is not extracted
```

Getting archive filenames from STDIN

Like `cpio`, pax can read filenames from standard input (`stdin`). This provides great flexibility - for example, a `find(1)` command may select files/directories in ways pax can't do itself. In **write** mode (creating an archive) or **copy** mode, when no filenames are given, pax expects to read filenames from standard input. For example:

```
# Back up config files changed less than 3 days ago
find /etc -type f -mtime -3 | pax -x ustar -w -f /backups/etc.tar

# Copy only the directories, not the files
mkdir /target
find . -type d -print | pax -r -w -d /target

# Back up anything that changed since the last backup
find . -newer /var/run/mylastbackup -print0 |
    pax -0 -x ustar -w -d -f /backups/mybackup.tar
touch /var/run/mylastbackup
```

The `-d` option tells pax `not` to recurse into directories it reads (`cpio` -style). Without `-d` , pax recurses into all directories (`tar` -style).

Note: the `-0` option is not standard, but is present in some implementations.

From tar to pax

`pax` can handle the `tar` archive format, if you want to switch to the standard tool an alias like:

```
alias tar='echo USE PAX, idiot. pax is the standard archiver!; # '
```

in your `~/.bashrc` can be useful 😀.

Here is a quick table comparing (GNU) `tar` and `pax` to help you to make the switch:

TAR	PAX	Notes
`tar xzvf file.tar.gz`	`pax -rvz -f file.tar.gz`	`-z` is an extension, POSIXly: `gunzip <file.tar.gz | pax -rv`
`tar czvf archive.tar.gz path …`	`pax -wvz -f archive.tar.gz path …`	`-z` is an extension, POSIXly: `pax -wv path | gzip > archive.tar.gz`
`tar xjvf file.tar.bz2`	`bunzip2 <file.tar.bz2 | pax -rv`	
`tar cjvf archive.tar.bz2 path …`	`pax -wv path | bzip2 > archive.tar.bz2`	
`tar tzvf file.tar.gz`	`pax -vz -f file.tar.gz`	`-z` is an extension, POSIXly: `gunzip <file.tar.gz | pax -v`

`pax` might not create ustar (`tar`) archives by default but its own pax format, add `-x ustar` if you want to ensure pax creates tar archives!

Implementations

- AT&T AST toolkit (http://www2.research.att.com/sw/download/) | manpage (http://www2.research.att.com/~gsf/man/man1/pax.html)
- Heirloom toolchest (http://heirloom.sourceforge.net/index.html) | manpage (http://heirloom.sourceforge.net/man/pax.1.html)
- OpenBSD pax (http://www.openbsd.org/cgi-bin/cvsweb/src/bin/pax/) | manpage (http://www.openbsd.org/cgi-bin/man.cgi?query=pax&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=
- MirBSD pax (https://launchpad.net/paxmirabilis) | manpage (https://www.mirbsd.org/htman/i386/man1/pax.htm) - Debian bases their package

upon this.
- SUS pax specification (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/pax.html)

Discussion

howto/pax.txt Last modified: 2015/08/09 04:55 by bill_thomson