GNU sharutils, version 4.1.9

A set of shell archiver utilities
Edition 4.1.9, 28 October

Jan Djärv
Francois Pinard

Short Contents

1 Introduction to both programs 1
2 Invoking the `shar` program 2
3 Invoking the `unshar` program 7
4 Miscellaneous considerations 8

Table of Contents

1 Introduction to both programs 1

2 Invoking the `shar` program 2
- 2.1 Selecting files 2
- 2.2 Splitting output 2
- 2.3 Controlling the shar headers 3
- 2.4 Selecting how files are stocked 4
- 2.5 Protecting against transmission errors 5
- 2.6 Producing different kinds of shars 5

3 Invoking the `unshar` program 7

4 Miscellaneous considerations 8

1 Introduction to both programs

GNU `shar` makes so-called shell archives out of many files, preparing them for transmission by electronic mail services. A *shell archive* is a collection of files that can be unpacked by `/bin/sh`. A wide range of features provide extensive flexibility in manufacturing shars and in specifying shar *smartness*. For example, `shar` may compress files, uuencode binary files, split long files and construct multi-part mailings, ensure correct unsharing order, and provide simplistic checksums. See Chapter 2 [shar invocation], page 2.

GNU `unshar` scans a set of mail messages looking for the start of shell archives. It will automatically strip off the mail headers and other introductory text. The archive bodies are then unpacked by a copy of the shell. `unshar` may also process files containing concatenated shell archives. See Chapter 3 [unshar invocation], page 7.

GNU `shar` has a long history. All along this long road, numerous users contributed various improvements. The file '`THANKS`', from the GNU `shar` distribution, contain all names still having valid email addresses, as far as we know.

Please help me getting the history straight, for the following information is approximative. James Gosling wrote the public domain `shar 1.x`. William Davidsen rewrote it as `shar 2.x`. Warren Tucker brought modifications and called it `shar 3.x`. Richard Gumpertz maintained it until 1990. Francois Pinard, from the public domain `shar 3.49`, made `GNU shar 4.x`, in 1994. Some modules and other code sections were freely borrowed from other GNU distributions, bringing this `shar` under the terms of the GNU General Public License.

Your feedback helps us to make a better and more portable product. Mail suggestions and bug reports (including documentation errors) for these programs to '`bug-gnu-utils@prep.ai.mit.edu`'.

2 Invoking the `shar` program

The format of the `shar` command is one of:

```
shar [ option ] ... file ...
shar -S [ option ] ...
```

In the first form, the file list is given as command arguments. In the second form, the file list is read from standard input. The resulting archive is sent to standard output unless the `-o` option is given.

Options can be given in any order. Some options depend on each other: the `-o` option is required if the `-l` or `-L` option is used. The `-n` option is required if the `-a` option is used. Also see `-V` below.

Some options are special purpose:

`--help` Print a help summary on standard output, then immediately exits.

`--version`
: Print the version number of the program on standard output, then immediately exits.

`-q`
`--quiet` Verbose *off* at `shar` time. Messages are usually issued on standard error to let the user follow the progress, while making the archives. This option inhibits these messages.

2.1 Selecting files

`-p`
`--intermix-type`
: Allow positional parameter options. The options `-M`, `-B`, `-T`, `-z` and `-Z` may be embedded, and files to the right of the option will be processed in the specified mode. Without the `-p` option, embedded options would be interpreted as file names. See Section 2.4 [Stocking], page 4 for more information on these options.

`-S`
`--stdin-file-list`
: Read list of files to be packed from the standard input rather than from the command line. Input must be one file name per line. This switch is especially useful when the command line will not hold the list of files to be packed. For example:

```
find . -type f -print | shar -S -o /tmp/big.shar
```

If `-p` is specified on the command line, then the options `-M`, `-B`, `-T`, `-z` and `-Z` may be included in the standard input (on a line separate from file names). The maximum number of lines of standard input, file names and options, may not exceed 1024.

2.2 Splitting output

`-o` *`prefix`*
`--output-prefix=`*`prefix`*
: Save the archive to files '*`prefix`*`.01`' through '*`prefix.nnn`*' instead of standard output. This option *must* be used when the `-l` or the `-L` switches are used.

When *prefix* contains any '`%`' character, *prefix* is then interpreted as a `sprintf` format, which should be able to display a single decimal number. When *prefix* does not contain such a '`%`' character, the string '`.%02d`' is internally appended.

`-l` *`size`*
`--whole-size-limit=`*`size`*
: Limit the output file size to *size* times 1024 bytes but don't split input files. This allows the recipient of the shell archives to unpack them in any order.

`-L` *`size`*
`--split-size-limit=`*`size`*
: Limit output file size to *size* times 1024 bytes and split files if necessary. The archives created with this option must be unpacked in the correct order. If the recipient of the shell archives wants to put all of them in a single folder, she shall save them in the correct order for `unshar`, used with option `-e`, to unpack them all at once. See Chapter 3 [unshar invocation], page 7.

  For people used to saving all the shell archives into a single mail folder, care must be taken to save them in the appropriate order. For those having the appropriate tools (like Masanobu Umeda's `rmailsort` package for GNU Emacs), shell archives can be saved in any order, then sorted by increasing date (or send time) before massive unpacking.

2.3 Controlling the shar headers

`-n` *`name`*
`--archive-name=`*`name`*
: Name of archive to be included in the header of the shar files. Also see the `-a` switch further down.

`-s` *`address`*
`--submitter=`*`address`*
: The `-s` option allows for overriding the email address for the submitter, for when the default is not appropriate. The automatically determined address looks like '*`username@hostname`*'.

`-a`
`--net-headers`
: Allows automatic generation of headers:

```
Submitted-by: address
Archive-name: name/partnn
```

  The *name* must be given with the `-n` switch. If name includes a '`/`', then '`/part`' isn't used. Thus '`-n xyzzy`' produces:

```
xyzzy/part01
xyzzy/part02
```

  while '`-n xyzzy/patch`' produces:

```
xyzzy/patch01
xyzzy/patch02
```

  and '`-n xyzzy/patch01.`' produces:

```
xyzzy/patch01.01
xyzzy/patch01.02
```

`-c`
`--cut-mark`
: Start the shar with a cut line. A line saying '`Cut here`' is placed at the start of each output file.

2.4 Selecting how files are stocked

`-T`
`--text-files`
: Treat all files as text, regardless of their contents.

`-B`
`--uuencode`
: Treat all files as binary, use `uuencode` prior to packing. This increases the size of the archive. The recipient must have `uudecode` in order to unpack.

> Use of `uuencode` is not appreciated by many on the net, because people like to readily see, by mere inspection of a shell archive, what it is about.

`-M`
`--mixed-uuencode`
: Mixed mode. Automatically determine if the files are text or binary and archive correctly. Files found to be binary are uuencoded prior to packing. This option is selected by default.

For a file is considered to be a text file, instead of a binary file, all the following should be true simultaneously:

1. The file does not contain any ASCII control character besides ⟨BS⟩ (backspace), ⟨HT⟩ (horizontal tab), ⟨LF⟩ (new line) or ⟨FF⟩ (form feed).
2. The file does not contains a ⟨DEL⟩ (delete).
3. The file contains no character with its eighth-bit set.
4. The file, unless totally empty, terminates with a ⟨LF⟩ (newline).
5. No line in the file contains more than 200 characters. For counting purpose, lines are separated by a ⟨LF⟩ (newline).

`-z`
`--gzip`
: Use `gzip` and `uuencode` on all files prior to packing. The recipient must have `uudecode` and `gzip` (used with `-d`) in order to unpack.

Usage of `-z` in net shars will cause you to be flamed off the earth.

`-g` *`level`*
`--level-for-gzip=`*`level`*
: When doing compression, use `-`*`level`* as a parameter to `gzip`. The `-g` option turns on the `-z` option by default. The default value is 9, that is, maximum compression.

`-Z`
`--compress`
: Use `compress` and `uuencode` on all files prior to packing. The recipient must have `uudecode` and `compress` (used with `-d`) in order to unpack. Option `-C` is a synonymous for `-Z`, but is deprecated.

Usage of `-Z` in net shars will cause you to be flamed off the earth.

`-b` *`bits`*
`--bits-per-code=`*`bits`*
: When doing compression, use `-bx` as a parameter to `compress`. The `-B` option turns on the `-Z` option by default. The default value is 12, foreseeing the memory limitations of some `compress` programs on smallish systems, at `unshar` time.

2.5 Protecting against transmission errors

Transmission of shell archives is not always free of errors. So one should make consistency checks on the receiving site. A very simple (and unreliable) method is running the UNIX `wc` tool on the output file. This can report the number of characters in the file.

As one can guess this does not catch all errors. Especially changing of a character value does not change the computed check sum. To achieve this goal better method were invented and standardized. One very strong is MD5 (MD = message digests). This is standardized in RFC 1321. The produced shell scripts do not force the `md5sum` program to be installed on the system. This is necessary because it is not yet part of every UNIX. The program is however not necessary for producing the shell archive.

`-w`
`--no-character-count`
: Do *not* check with '`wc -c`' after unpack. The default is to check.

`-D`
`--no-md5-digest`
: Do *not* check with '`md5sum`' after unpack. The default is to check.

`-F`
`--force-prefix`
: Prepend the prefix character to every line even if not required. This option may slightly increase the size of the archive, especially if `-B` or `-Z` is used. Normally, the prefix character is '`X`'. If the parameter to the `-d` option starts with '`X`', then the prefix character becomes '`Y`'.

`-d` *`string`*
`--here-delimiter=`*`string`*
: Use *string* to delimit the files in the shar instead of '`SHAR_EOF`'. This is for those who want to personalize their shar files.

2.6 Producing different kinds of shars

`-V`
`--vanilla-operation`
: This option produces *vanilla* shars which rely only upon the existence of `echo`, `test` and `sed` in the unpacking environment.

  The `-V` disables options offensive to the *network cop* (or *brown shirt*). It also changes the default from mixed mode `-M` to text mode `-T`. Warnings are produced if option `-B`, `-z`, `-Z`, `-p` or `-M` is specified (any of which does or might require `uudecode`, `gzip` or `compress` in the unpacking environment).

`-P`
`--no-piping`
: In the shar file, use a temporary file to hold the file to `uudecode`, instead of using pipes. This option is mandatory when you know the unpacking `uudecode` is unwilling to merely read its standard input. Richard Marks wrote what is certainly the most (in)famous of these, for MSDOS :-).

  (Here is a side note from the maintainer. Why isn't this option the default? In the past history of `shar`, it was decided that piping was better, surely because it is less demanding on disk space, and people seem to be happy with this. Besides, I think that the `uudecode` from Richard Marks, on MSDOS, is wrong in refusing to handle `stdin`. So far that I remember, he has the strong opinion that a program without any parameters should give its `--help` output. Besides that, should I say, his uuencode and uudecode programs are full-featured, one of the most complete set I ever saw. But Richard will not release his sources, he wants to stay in control.)

`-x`
`--no-check-existing`
: Overwrite existing files without checking. If neither `-x` nor `-X` is specified, when unpacking itself, the shell archive will check for and not overwrite existing files (unless `-c` is passed as a parameter to the script when unpacking).

`-X`
`--query-user`
: Interactively overwrite existing files.

  Use of `-X` produces shars which *will* cause problems with some unshar-style procedures, particularily when used together with vanilla mode (`-V`). Use this feature mainly for archives to be passed among agreeable parties. Certainly, `-X` is *not* for shell archives which are to be submitted to Usenet or other public networks.

  The problem is that unshar programs or procedures often feed '`/bin/sh`' from its standard input, thus putting '`/bin/sh`' and the shell archive script in competition for input lines. As an attempt to alleviate this problem, shar will try to detect if '`/dev/tty`' exists at the receiving site and will use it to read user replies. But this does not work in all cases, it may happen that the receiving user will have to avoid using unshar programs or procedures, and call `/bin/sh` directly. In vanilla mode, using '`/dev/tty`' is not even attempted.

`-m`
`--no-timestamp`
: Avoid generating touch commands to restore the file modification dates when unpacking files from the archive.

  When the timestamp relationship is not preserved, some files like '`configure`' or '`*.info`' may be uselessly remade after unpacking. This is why, when this option is not used, a special effort is made to restore timestamps,

`-Q`
`--quiet-unshar`
: Verbose *off* at unshar time. Disables the inclusion of comments to be output when the archive is unpacked.

`-f`
`--basename`
: Use only the last file name component of each input file name, ignoring any prefix directories. This is sometimes useful when building a shar from several directories, or another directory. If a directory name is passed to shar, the substructure of that directory will be restored whether `-f` is specified or not.

3 Invoking the `unshar` program

The format of the `unshar` command is:

```
unshar [ option ] ... [ file ... ]
```

Each *file* is processed in turn, as a shell archive or a collection of shell archives. If no files are given, then standard input is processed instead.

Options:

`--version`
: Print the version number of the program on standard output, then immediately exits.

`--help`
: Print an help summary on standard output, then immediately exits.

`-d` *`directory`*
`--directory=`*`directory`*
: Change directory to *directory* before unpacking any files.

`-c`
`--overwrite`
`-f`
`--force`
: Passed as an option to the shar file. Many shell archive scripts (including those produced by `shar` 3.40 and newer) accepts a `-c` argument to indicate that existing files should be overwritten.

: The option `-f` is provided for a more unique interface. Many programs (such as `cp` and `mv`) use this option to trigger the very same action.

`-e`
`--exit-0`
: This option exists mainly for people who collect many shell archives into a single mail folder. With this option, `unshar` isolates each different shell archive from the others which have been put in the same file, unpacking each in turn, from the beginning of the file towards its end. Its proper operation relies on the fact that many shar files are terminated by a '`exit 0`' at the beginning of a line.

: Option `-e` is internally equivalent to `-E "exit 0"`.

`-E` *`string`*
`--split-at=`*`string`*
: This option works like `-e`, but it allows you to specify the string that separates archives if '`exit 0`' isn't appropriate.

: For example, noticing that most '`.signatures`' have a '`--`' on a line right before them, one can sometimes use '`--split-at=--`' for splitting shell archives which lack the '`exit 0`' line at end. The signature will then be skipped altogether with the headers of the following message.

4 Miscellaneous considerations

Here is a place-holder for many considerations which do not fit elsewhere, while not worth a section for themselves.

Be careful that the output file(s) are not included in the inputs or `shar` may loop until the disk fills up. Be particularly careful when a directory is passed to `shar` that the output files are not in that directory (or a subdirectory of that directory).

When a directory is passed to `shar`, it may be scanned more than once, to conserve memory. Therefore, one should be careful to not change the directory contents while `shar` is running.

No attempt is made to restore the protection and modification dates for directories, even if this is done by default for files. Thus, if a directory is given to `shar`, the protection and modification dates of corresponding unpacked directory may not match those of the original.

Use of the `-M` or `-B` options will slow down the archive process. Use of the `-z` or `-Z` options may slow the archive process considerably.

Let us conclude by a showing a few examples of `shar` usage:

```
shar *.c > cprog.shar
shar -Q *.[ch] > cprog.shar
shar -B -l28 -oarc.sh. *.arc
shar -f /lcl/src/u*.c > u.sh
```

The first shows how to make a shell archive out of all C program sources. The second produces a shell archive with all '`.c`' and '`.h`' files, which unpacks silently. The third gives a shell archive of all uuencoded '`.arc`' files, into files '`arc.sh.01`' through to '`arc.sh.`*nnn*'. The last example gives a shell archive which will use only the file names at unpack time.