Using GNU tar 1.10 on the NeXT

Pierce E. Cantrell *

August 13, 1991

I. Introduction

The original versions of the level-0 and level-1 scripts delivered with GNU tar 1.10 assume that these scripts will be run as root on the computer that physically has the tape drive attached. The scripts use rsh (remote shell) to run GNU tar on a series of remote workstations to backup one or more of each workstation's file systems to the local tape drive using rmt. This is a very good approach for backing up a collection of workstations that each have local storage. It is only necessary to have GNU tar installed on each of the workstations; it is not necessary to have any of these file systems exported using NFS. With this method, you only need to enter the machine with the tape drive into each of the workstations' /etc/hosts.equiv file and to enter root into each of the workstations' /.rhosts file.

The approach described above looks like a good long term solution if you have, for example, and Exabyte or DAT drive available. In our case, however, both of our Exabyte drives on the two Mips file servers are left on-line for automatic backups using dump instead of tar. The only other tape drive was a 120 MB cartridge that, because of its size, is not very suitable for the backups described above. What we decided to do was to modify the level-0 and level-1 scripts so that they could be run from a NeXT, or other workstation, without a tape drive to a remote tape drive. This was a trivial modification to the original scripts. In addition, we wanted to be able to use floppy disks for level-1 backups, and we wanted to add a level-2 tar to do daily backups to floppy disk. With some minor modifications to the original level-0 and level-1 scripts, you could use them with the level-1-floppy and level-2 scripts described in this document (add a line to copy

*The author is with the Department of Electrical Engineering, Texas A&M University, College Station, Texas 77843–3128. Telephone (409) 845–7441. Electronic Mail: `cantrell@ee.tamu.edu`

the temp.level-0 file and the temp.level-1 file to $fsname.level-1-floppy in the level-0 and level-1 scripts, respectively. You will also have to use the BACKUP_DIRS method and not the BACKUP_FILES method. The original scripts also used the `+sparse` option which we chose not to use.).

In Section II we describe how to use GNU tar version 1.10 to backup and restore directories and files using a remote mag tape and a local floppy disk on the NeXT. We assume you have the the following scripts as modified by Cantrell (cantrell@ee.tamu.edu): level-0, level-1, level-1-floppy, level-2, backup-specs-tape, backup-specs-floppy-1, backup-specs-floppy-2. There is also a script that writes directly to the floppy /dev/fd0a, level-1-fd0a and backup-specs-1-fd0a, but there is no advantage to using this over level-1-floppy (the latter version writes using the file system, and it is easier to use and faster). In Section III we describe how to use GNU tar for non-incremental backups. There are seven subsections that describe all of the most common uses.

II. Using GNU tar for Incremental Backups and Restores

In this section, we describe how to configure the scripts for making incremental tars, running the incremental backups, and how to recover directories and files from the incremental tar tapes and floppies.

A. *Configuration of the level-x scripts*

There are four shell scripts for performing incremental dumps: level-0, level-1, level-1-floppy, and level-2. These scripts read configuration parameters from the files backup-specs-tape (for both level-0 and level-1), backup-specs-floppy-1 (for level-1-floppy) and backup-specs-floppy-2 (for level-2). To change these scripts for your system, you will have to adapt the backup-specs-∗ files.

Configuring backup-specs-tape

The modified versions of the level-0 and level-1 scripts allow you to run them on a machine that does *not* have have a tape drive (see Section I for a description of how the original versions of the scripts worked). Change the variable TAPE_FILE to be the tape drive on the remote system. Change variable TAPE_HOST to be the internet name of the remote system with the tape drive. The variable USERID is the user id on system

TAPE_HOST that will be used by rsh (remote shell) for executing the rmt (remote tape server) program. In order for this to work, the remote system will have to have a .rhosts file in the home directory for USERID with an entry for root, and USERID on system TAPE_HOST will have to have the appropriate privilege to access the tape drive. Your machine will also have to be in TAPE_HOST's /etc/hosts.equiv file. An alternative to this is to have your workstation included in /.rhosts and to modify the level-0 and level-1 scripts by eliminating all occurrences of "USERID@". The system manager of the machine with the tape drive may not want to do this, however, because it gives you root privileges on that system if you have root privileges on your own workstation.

Change variables BACKUP_DIRS and BACKUP_FILES as appropriate. For what I wanted to do (routinely backup only those directories that I have made changes to), the BACKUP_FILES approach seemed the best. Each entry in BACKUP_DIRS causes tar to be executed. Therefore, you generate a separate gnu dump file in /etc/tar-backup and a separate tar file for each entry in BACKUP_DIRS. If you are going to specify "/" or an entire user file system, the BACKUP_DIRS method seems appropriate. If you are going to list 13 separate directories as I did, however, the BACKUP_DIRS method would result in 13 tar files; whereas, the BACKUP_FILES method results in only one tar file. It takes less time to dump to floppy when using the BACKUP_FILES approach in the example above because tar is only run once. In either case, you must be consistent between the tape and floppy tars (i.e., both use BACKUP_DIRS or both use BACKUP_FILES) because the `+listed-incremental` option of tar needs to read and write the gnu dump file in /etc/tar-backup. I have not tested the BACKUP_DIRS code very well – so beware.

Configuring backup-specs-floppy-1

The variable FD_FILE needs to be the "name" of the floppy disk (this will become the directory that it is mounted as). All of the disks you use in a multi-volume dump will need to have this same name. (I suppose it would be better to change the level-1-floppy script to mount the disk so that tar does not die if you don't have the first floppy mounted). The variable FD_SIZE is equal to the number of one KB blocks (K=1024) that tar will write before it asks for the next volume. This number of K Bytes needs to be able to fit on the disk. For a 1.44 MB floppy, FD_SIZE=1200 results in a disk that shows 97% full after tar writes to the disk, and FD_SIZE=2600 shows 99% full after the tar. Whatever you did above for tape backups with respect to BACKUP_DIRS and BACKUP_FILES, you need to do the same thing here.

Configuring backup-specs-floppy-2

Set FD_FILE and be consistent with respect to using either BACKUP_DIRS or BACKUP_FILES. I have chosen to make level-2 backup only a subset of the files in the level-1 dump, so that a number of the backups will fit on one floppy disk. All I have chosen to backup is my own personal directory and /private/etc. Since level-2 dumps are compressed tar files, they have to fit on one volume. Clearly, you can loose some things if something bad happens and the level-2 tar is not identical to levels 0 and 1. You will need to experiment here and see what you think.

B. Running the level-x scripts

To run any of the four level$-x$ scripts, you need to be logged in as root and then type, for example,

```
level-0 now
```

to perform a level-0 backup right now. Similarly, you can run level-1, level-1-floppy and level-2 tars. A log file will be sent to userid ADMINSTRATOR specified in the backup-specs file, and the log file will also be left in the directory from which you run level$-x$. You can also specify a time to run the script; I have not tested this option. The level-0, level-1, and level-1-floppy dumps are multi-volume. If the data does not fit on one tape or floppy, you will be requested to insert a new one and hit return. For floppy disks, I just drag the floppy to the recycler and put in a new one.

The incremental dumps are done using the `+listed-incremental` mode in GNU tar. The listed-incremental mode uses a file written by the next lower numbered script, which is stored in the /etc/tar-backup directory, to avoid dumping directories that have not changed. These files are only needed for the backup and not the restore. If one of these files gets deleted, you will dump a lot more than you expect, but nothing will be lost. The level-0 tar will write one of these files but not read one.

I suggest you run a level-0 dump at least once a month. In an ideal world, you would keep these forever. I suggest a minimum of one year. Every two weeks would even be better. I would then run a level-1 dump to either tape or floppy (depending on how much stuff you have to dump) once a week. You only need one of these to recover since a level-1 tar dumps everything since the last level-0, but I would keep the current and one previous in case you have media problems on the current level-1 media (You will also need to save the level-2 immediately before the current level-1 to get everything back. If you do like I am and have level-2 only be a subset of level-1, you could loose a week's worth of work this way). Since your level-0 media can also go bad, it would be nice to keep the last level-1 dump from the previous month for at least a month (in order to have everything this would need to be run right before the level-0 dump). Level-2 dumps go only to floppy disk as a compressed tar file.

C. Full restore of directories from incrementals

In restoring from incrementals, remember that each level$-n$ dump stores everything that is new or changed since the last level$-(n-1)$ dump. Therefore, you should at most have to read three tars to get everything back. The way the incremental dumps work is that they write out a special record to the tar tape or floppy (not compatible with other versions of tar) that lists all the files in a directory and whether or not they are included in the tar file.

In the scenario below, foo/bar is a directory, file, or list of directories and files that you want to restore. GNU tar removes leading "/" 's from directory or filenames when it writes the archive. That is why the restore cd's to the root directory. If you eliminate the `cd /` from below, you could restore the files under the current working directory. This feature allows you to examine what is in an archive before you actually restore everything (This is also a good way to test your dump and restore procedures without actually having to do the real thing). You will have to be logged in as root. If you don't list any directories or files to restore, everything in the tar file will be restored. A common error is to put the leading "/" on in the file list; tar will *not* find `/foo/bar`!

In the examples below, wherever you see `cantrell@ee:/dev/rmt/ctape0`, replace the string with the userid, hostname, and tape drive name that is appropriate for your system. For floppy disk tars, you will need the "name" of the floppy disk and the filename from which to restore. If you used the BACKUP_FILES method, the level-1-floppy files will be named something like misc-level-1-Jun-5-1991.tar and the level-2 files will be named something like misc-level-2-Jun-5-1991.tar.Z (these are compressed).

In step 1 below, the `+incremental` switch is used. If you are restoring a directory from scratch that is currently nonexistent, the presence or absence of the `+incremental` switch will not have an effect on the restore. If there are files in the directory, however, they will be deleted if they are not found in the level-0 tar.

Step 1

From your most recent level-0 dump tape, do the following:

```
cd /
/usr/local/bin/tar -x +multi-volume +block=64 \
+incremental \
-f cantrell@ee:/dev/rmt/ctape0 \
foo/bar
```

Step 2

From your most recent level-1 tape or floppy, do the following for tape

```
cd /
/usr/local/bin/tar -x +multi-volume +block=64 \
+incremental \
-f cantrell@ee:/dev/rmt/ctape0 \
 foo/bar
```

or this for floppy

```
cd /
/usr/local/bin/tar -x +multi-volume +block=20 \
+incremental \
-f /Floppy_directory/filename.tar \
foo/bar
```

Step 3

From your most recent level-2 floppy, do the following:

```
/usr/local/bin/tar -x -z +block=20
+incremental \
-f /Floppy_directory/filename.tar.Z \
foo/bar
```

D. *Single file restoration when you know the last modification date*

The method described above will work for a single file as well as a directory. If you have an idea or know when the last time you modified a single file, however, it will probably be faster to extract that single file from one of the incremental dumps. If you know, for example, that you modified a file yesterday and then accidently deleted it today, you can extract that single file from the level-2 incremental dump from the day before. To extract from a level-2 dump see Section III D below. To extract from a level-1 dump see III A or III C. If you just want to see what is on a tar tape or disk file, use the `-t` (list) option instead of the `-x` option. You can also add the `-v` (verbose) option for more output.

III. Using GNU tar for Non-incremental Backups and Restores

In the following sections, `/foo/bar` (or `foo/bar` for relative path names) refers to a single file, a directory, or a list of files or directories. In sections A and B, you will need to replace `cantrell@ee:/dev/rmt/ctape0` with the userid on the machine with the tape, tape host, and tape name. If you want to list what is on a tar instead of extracting, use the `-t` option instead of the `-x` option. Adding `-v` to either results in more verbose output. A tar to tape is shown using a block size of 64. The typical number for streaming tape drives is more often 126, but the Mips 3230 has a maximum size of 64; use whatever you like, but I would test it first.

A. *Multi-volume to Tape*

The dump instructions are followed by the restore instructions. You don't have to set the label, but if the paper label falls off the tape you will appreciate knowing what is there and when it was written. Multi-volume dumps are not compatible with other versions of tar. Only GNU tar supports this useful feature. By default, GNU tar strips the leading "/" from a directory or filename when it writes the archive. To restore the file, you will need to `cd` to the proper parent directory. If you dump it with an absolute path name, then remember to cd to / to restore. This relative path name is a nice feature in that you can restore an entire directory chain somewhere else to look at it first. (GNU tar also has a -P switch that will allow absolute path names)

Dump:

```
set date=`date`
```

```
/usr/local/bin/tar -c +multi-volume +block=64 \
+label="Something Descriptive at $date" \
-f cantrell@ee:/dev/rmt/ctape0 \
 /foo/bar
```

When restoring a single file or directory, you can begin with the volume that the file starts on if you know it. For example, if you know that a file or directory begins on volume 3, you can start with this volume without first reading through volumes 1 and 2. If you don't know where the file is, begin with volume 1. It is not likely that you will know the volume number where a file or directory begins unless you have used the `-t` option to list the contents of a tar file or tape. It is common that the last file on a volume will be split between the current and following volumes. To restore this split file, you must begin with the volume that the file starts on. You must also use the `+multi-volume` (`-M`) switch to recover a file split between volumes.

Restore:

```
/usr/local/bin/tar -x +multi-volume +block=64 \
-f cantrell@ee:/dev/rmt/ctape0 \
foo/bar
```

B. Compressed and Uncompressed Single Volume to Tape

You cannot do multi-volume compressed dumps. Again, labels are a good idea, but they are *not* compatible with other versions of tar. Therefore, if you need to transport something to another system that does not have GNU tar, don't use a label. If you want and uncompressed tar, don't include the `-z` option.

Dump:

```
set date=`date`
/usr/local/bin/tar -c -z +block=64 \
+label="Something Descriptive at $date" \
-f cantrell@ee:/dev/rmt/ctape0 \
/foo/bar
```

Restore:

```
/usr/local/bin/tar -x -z +block=64 \
```

```
-f cantrell@ee:/dev/rmt/ctape0 \
foo/bar
```

C. Multi-volume to Floppy (Using the file system)

This is the normal way to do a multi-volume tar to floppy if you don't need to transport the output to another system. If you need to move from a NeXT to a Sun, see F or G below. Recall that multi-volume tar is not compatible with other versions of tar. For a 1.44 MB floppy, a tape-length of 1200 (i.e., 1200 K Bytes where K=1024) results in 97% of the floppy being used after the tar. For the 2.88 MB disks, a tape-length of 2600 results in 99% of the floppy used.

Dump:

```
set date=`date`
/usr/local/bin/tar -c +multi-volume +block=20 \
+label="Something Descriptive at $date" \
+tape-length=1200 \
-f /Floppy_Directory/filename.tar \
/foo/bar
```

When restoring a single file or directory, you can begin with the volume that the file starts on if you know it. For example, if you know that a file or directory begin on volume 3, you can start with this volume without first reading through volumes 1 and 2. If you don't know where the file is, begin with volume 1. It is not likely that you will know the volume number where a file or directory begins unless you have used the `-t` option to list the contents of a tar file or tape. It is common that the last file on a volume will be split between the current and following volumes. To restore this split file, you must begin with the volume that the file starts on. You must use the `+multi-volume` (`-M`) switch to recover a file split between volumes. There is no `+tape-length` switch since it is not used when extracting or listing an archive (it does no harm to include it, however).

Restore:

```
/usr/local/bin/tar -x +multi-volume +block=20 \
-f /Floppy_Directory/filename.tar \
foo/bar
```

D. Compressed Single Volume to Floppy or Hard Disk

To be compatible with other tars, we won't use a `+label` here, but you can use one if you are not worried about compatibility. Since this is a single volume tar, there is no distinction between a dump to floppy or to a file on the hard disk. We just use the default number of blocks, which is normally equal to 20.

Dump:

```
/usr/local/bin/tar -c -z \
-f /Somedirectory/filename.tar.Z \
/foo/bar
```

Restore:

```
/usr/local/bin/tar -x -z \
-f /Somedirectory/filename.tar.Z \
foo/bar
```

E. Single Volume to Floppy or Hardisk

This is identical to D above without the `-z` switch. Again, this will be compatible with other tars.

Dump:

```
/usr/local/bin/tar -c \
-f /Somedirectory/filename.tar \
/foo/bar
```

Restore:

```
/usr/local/bin/tar -x \
-f /Somedirectory/filename.tar \
foo/bar
```

F. Multi-volume to /dev/fd0a (floppy)

The tar described in this and the following section does a direct write to the floppy disk without having a Unix file system created on the floppy. A tar made in this way will not mount properly if inserted into the NeXT floppy drive with the automounter in operation.

In fact, if you accidently insert a write enabled disk on which you have previously written a tar using the method of this section, the data will be partially destroyed. Therefore, the recommendation is to write-protect the floppy immediately after you do the tar.

The only reason for running tar's like these is to be able to transport the floppy tar between, for example, a NeXT and a Sun. Formerly in tar version 1.09, a tar to /dev/fd0a was the only type of multi-volume floppy tar that could be performed. The `+tape-length` option is new to tar version 1.10. You cannot transport a tar made using the unix filesystem because the Sun and NeXT floppy implementations are not compatible.

The default block size of 20 will not work on the NeXT for this type of tar because of the error code returned from a partially successful read at the end of the disk. I have tested the block size of 4 on both 1.44 and 2.88 MB disks. I have been told, but not personally tested, that it is necessary to do this same thing on a Sun.

There are several steps in this operation. You must be careful not to insert the floppy disk until you get a window prompt telling you to insert the disk or things will not work correctly. You will normally need to be super user to access /dev/fd0a.

1. Format all the floppy disks you will need in advance. Be sure and discard any that don't format properly.

2. Run the dump or restore sequence give below. Don't insert a floppy until you see the pop-up window that instructs you to load the disk.

3. When you receive the message that tells you to load the next volume, type the following statements to eject the disk. After typing the last carriage return <CR> in the sequence below, you will see the pop-up window telling you to insert the new disk.

   (a) `! <cr>`

   (b) `/usr/etc/disk -e /dev/rfd0a <cr>`

   (c) `exit <cr>`

   (d) `<cr>`

4. Repeat the step above until the tar is complete.

Dump:

```
set date=`date`
/usr/local/bin/tar -c +multi-volume \
+block=4 \
+label="Something Descriptive at $date" \
-f /dev/fd0a \
/foo/bar
```

When restoring a single file or directory, you can begin with the volume that the file starts on if you know it. For example, if you know that a file or directory begin on volume 3, you can start with this volume without first reading through volumes 1 and 2. If you don't know where the file is, begin with volume 1. It is not likely that you will know the volume number where a file or directory begins unless you have used the `-t` option to list the contents of a tar file or tape. It is common that the last file on a volume will be split between the current and following volumes. To restore this split file, you must begin with the volume that the file starts on. You must use the `+multi-volume` (`-M`) switch to recover a file split between volumes.

Restore:

```
/usr/local/bin/tar -x +multi-volume \
+block=4 \
-f /dev/fd0a \
foo/bar
```

G. Single Volume to /dev/fd0a (Floppy)

The only difference here is that we are writing a single volume. You must still be careful not to insert the floppy until you receive the pop-up prompt to load the disk. See the section above for more details. I don't know whether or not this is compatible with other tars, but it might be. The `-b` notation is the standard tar notation for block size.

Dump:

```
/usr/local/bin/tar -c \
-b 4 \
-f /dev/fd0a \
/foo/bar
```

Restore:

```
/usr/local/bin/tar -x \
-b 4
-f /dev/fd0a \
foo/bar
```