ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/TC–97/30
3 Ottobre 1997

C. Strizzolo:

TAPEUTILS: A SET OF TOOLS FOR TAPES HANDLING ON DIGITAL UNIX. VERSION 1.0a

SIS–Pubblicazioni
dei Laboratori Nazionali di Frascati

INFN – Istituto Nazionale di Fisica Nucleare
Sezione di Trieste

INFN/TC–97/30
3 Ottobre 1997

TapeUtils

A set of tools for tapes handling on Digital Unix

Version 1.0a

Claudio Strizzolo
(I.N.F.N. Sezione di Trieste)

Abstract

This document describes the TapeUtils package, which includes some tools developed to improve and simplify tapes handling on Digital Unix systems.

TapeUtils includes a VMS-like tape units allocation/deallocation system and an enhanced version of Ed Bailey's popular juke-box loader control program, with improved security features.

Contents

1 Introduction

The TapeUtils package includes a set of tools that were developed to make tapes handling easier and safer on Digital Unix systems.

At present, the following tools are included in the package:

- `allocate/deallocate` - `allocate` provides a user with exclusive access to a (tape) device until he/she deallocates it (just like the `ALLOCATE` command on VMS operating system). This can be useful, for instance, when working with a tape loader that can be controlled through software, as in Unix there is no way to avoid an user to complete successfully a loading command, even if the unit is already in use by someone else (that is, the tape unit, even if in use, is unloaded and the new task is executed).
- `loader` - The package includes a modified version of the popular juke-box loader control program written by Ed Bailey. This modified version is able to work in combination with `allocate/deallocate`, with some more security features added.

2 Availability

The kit is available through FTP anonymous: `ftp://ftp.ts.infn.it/pub/unix/TapeUtils`.

The TapeUtils WWW home page is at the following URL: `http://www.ts.infn.it/computing/TapeUtils/`.

The latest version of the loader control program as developed by Ed Bailey on July, 1997, is available (as kindly requested by Ed Bailey, who changed his activity and does not work on this stuff any more) at the following URL: `ftp://ftp.ts.infn.it/pub/unix/loader-stuff.tar.gz`. You should not need it, as TapeUtils includes a compatible version with some improved features.

3 Installation

You must have superuser privileges to install TapeUtils.

1. Expand the kit
2. Edit the `TapeUtils.h` file to customize the following parameters:

   LOCK_FILE_PATH : The `allocate` tool creates a small lock file in order to lock access to each device. This parameter defines the path to the directory in which such lock files will be created.

   LIST_OF_DEVICES : This parameter defines the name of a file containing the list of devices that users will be allowed to allocate/deallocate.

   DAEMON_DELTA_TIME : The deallocation daemon (which deallocates devices allocated by no-more-existing processes, to avoid people forgetting allocated devices they do not use any more) scans devices at fixed intervals. Use this parameter to set the interval time length (in seconds).

LOADER_AUTH_FILES_PATH : This parameter defines the path to a directory containing the by-user authorization files to access the slots of a juke-box. Read section 5 for more information about the loader authorization system. If you do not mean to use authorizations on juke-boxes, or you do not have juke-boxes at all, just set this parameter to a non-existing path.

3. Build the SCSI/CAM libraries needed by the loader control program. This can be done by just executing the `make` command in the `scsilib` sub-directory.

   As described in the `loader`'s `README` file included below, in order to set the SCSI/CAM libraries correctly, you need to create a file named `/usr/local/etc/cam.lock` (use `touch`, for instance), with read/write permission for whoever will be using the loader program. Check also if a `/dev/cam` special file is available, with the same permissions. If not so, set it up appropriately.

4. Compile and build the executables. This can be done by just executing the `build` script:

   ```
   # ./build
   ```

   This script will compile the sources and build the executables into `/usr/local/bin`. If you prefer a different destination directory, just modify the `build` script before running it. The same applies if you do not mean to install any of the tools.

   The same script sets appropriate protections for allocate and deallocate images (in particular, it enables user ID setting on execution). *Do not change these settings!*

   Man pages are installed by default in `/usr/local/man`. Change the script appropriately if you prefer a different destination.

5. Create the file pointed by LIST_OF_DEVICES parameter. It should contain the list of devices that may be allocated, one per line, for instance:

   ```
   nrmt1h
   nrmt0h
   ```

   Only the devices listed in this file will be allowed to be allocated by the users. The file may be changed at any time in the future.

   Set the ownership of any device listed in the file to `root`, and grant access to the owner only. For instance:

   ```
   # chown root /dev/nrmt0h
   # chmod 700 /dev/nrmt0h
   ```

   This will make the devices available only by using the `allocate/deallocate` tools (well, `root` user excluded, of course).

6. If you mean to use the by-user authorization control system for the loader control program, read section 5.3 for more information about building the authorization files.

7. Start the `deallocd` daemon, for automatic deallocation of devices that people leaves allocated. For instance:

```
# /usr/local/bin/deallocd > /usr/local/log/deallocd.log 2>&1 &
```

As a suggestion, redirect the output to a log file, to keep trace of what happens.

You may include the daemon startup in your system startup scripts, to start it automatically every time your system boots.

8. If you mean to use the `loader` stuff, read carefully the `loader`'s `README` file which is included entirely in section 5, and follow the installation notes available there. In particular, you have to define a symbolic link to the `loader` executable for each loader you mean to support, like this one:

```
# ln -s /usr/local/bin/loader /usr/local/bin/nrmt0h-loader
```

4 Allocate/deallocate tools

The `allocate/deallocate` tools provide a user with exclusive access to a (tape) device until he/she deallocates it, just like the `ALLOCATE` command on VMS operating system.

4.1 Usage

If the installation completes successfully, the `allocate` and `deallocate` tools should be executable by any user with the following syntax:

`allocate` *device-name*
`deallocate` *device-name*

For instance:

```
allocate nrmt1h
```

Specifying the special character file (i.e. `/dev/nrmt1h`) is not allowed.

The `deallocate` command must be invoked by the same process which allocated the device.

4.2 How do they work?

When a device is not allocated by anyone, the device itself is owned by `root`, with no access by other users. This is set up during the installation. In this way, no one (except `root`) can access the device.

The `allocate` tool changes the owner of the device to the user executing `allocate` itself, keeping protections unchanged. This grants the user with exclusive access, because other users do not have the permissions to access the device. A lock file is created somewhere (the location is customized during the installation) to keep trace of who allocated what.

To summarize, when you try to allocate a device through the `allocate` command, the following steps are executed:

1. The device is searched in the list of allowed devices. If the list is not available, or the device is not listed there, the allocation fails. This step is performed to allow the system administrator to make only some devices available to the other users.

2. The tool checks if the device is allocated by another user, by looking for the lock file which corresponds to the device. If it is already allocated, the allocation fails.

3. The tool changes the ownership of `/dev/`*device-name* to the current userid, granting access to the current user only, and saves the user id and group id into the lock file.

After the allocation, the user can do his/her job safely: no one can interfere on the same device.

When the job is done, the user must deallocate the device, to make it available to other users. The `deallocate` tool just does the following:

1. The tool checks if the device was really allocated by the same process which is trying to deallocate it, by looking into the lock file. If not so, the deallocation fails.

2. The tool changes the ownership of `/dev/`*device-name* to `root` again, and removes the lock file.

4.3 Forcing deallocation

On VMS operating system, a device remains allocated until the user deallocates it or the process which allocated it terminates.

On Unix, the latest is not done automatically. This is potentially dangerous: a process might crash without deallocating a device, or the user might forget deallocating it.

To avoid problems, a daemon can be submitted to force deallocation for devices allocated by no-more-existent processes. The daemon (named `deallocd`) scans all the devices at fixed intervals, and deallocates them if the above condition happens. The length of the time interval is established by setting the DAEMON_DELTA_TIME parameter in the configuration file during the installation.

Read the installation notes for more information about submitting the daemon.

If needed, the system administrator may force a deallocation "by hand", just by performing the following steps:

1. Remove the lock file. The location is defined by the LOCK_FILE_PATH parameter in the configuration file during the installation. The name of the file is the same of the device.

2. Change the owner of the device character file (i.e. `/dev/`*device-name*) to `root`.

5 Loader control program

The tape loader control utility was written by Ed Bailey (bailey@niehs.nih.gov) on April 1995 and modified on July 1997 to fix some problems with Digital Unix 4.0. This tool makes you able to control a SCSI tape loader (juke-box) through software.

The original `README` file by Ed Bailey follows:

```
                                                    6 April 1995
Hello!

    You are now the proud owner of my tape loader control utility.  I wrote
```

this program to permit the amanda network backup software (highly recommended: check out ftp.cs.umd.edu, in /pub/amanda for the latest copy.) to directly control our DEC TZ877 seven-slot DLT loader. It's been in use now for over a month with nary a hiccup, so I thought I'd give it to others to break. ;-)

Random Notes:

- This program uses Wolfgang Barth's SCSI access library to talk to the loader via /dev/cam. The original version of Wolfgang's library can be obtained from: speckled.mpifr-bonn.mpg.de in /pub/scsi/scsilib.tar.gz. However, since Wolfgang wrote this code to run under DEC Ultrix, and we're running DEC OSF/1 (now called Digital Unix), some changes had to be made. They're minor, but I'm making available a modified version of scsilib.tar.gz for those that don't feel the need to hack it on their own...

- Makefile? We don't need no steenkin' Makefile! To build this program, Issue the following command:

  cc loader.c -o loader -L<libscsi-location> -lscsi

  Where <libscsi-location> is the location of libscsi.a.

- Speaking of locations of things, I copped out and simply made a directory called "loader-src" on the same level as the directory (called "scsilib") that held Wolfgang's code. The only artifact of this layout are the two #includes in loader.c that refer to "../scsilib/<whatever>". So if you set up things differently, those #includes are going to have to change.

- The program as written looks at the name it was invoked under in order to determine the name of the loader it is to control. The name of the program should look something like:

  foonly-loader

  Where "foonly" is the name of the tape device. This device is expected to reside in /dev, but this can be changed. So basically what it does is to grab the part of the name prior to "-loader", prepend "/dev/" to it, and then get the bus and target (aka SCSI id) from that device's minor number. This is probably completely OSF/1 dependent, so if it doesn't work on your non-OSF/1 machine, I told ya so!

  Anyway, I ended up plopping the executable "loader" in /usr/local/bin, and then symbolically linking nrmt0h-loader and nrmt0m-loader to it. This way, you can control different loaders with only one copy of the program. Amanda users: Note that the device name the program returns is what amanda will use for backup purposes, so you need to get the name just right, or your tape density/rewind stuff will be gronked.

- Wolfgang's library routines use a locking file to control access to

/dev/cam. In order to make *anything* that uses his routines work, you need to create a file called "cam.lock" in /usr/local/etc (touch works fine). This file will need read/write permission for whoever will be using the loader program. Can you move the location of the file? I don't see why not, but I didn't bother.

o As noted above, this program was written to control a DEC TZ877. If I was a betting person, I'd bet that it would work on a DEC TZ875, and the various OEM versions that Quantum (who bought the DLT business from DEC recently) produces. I tried to make the code as general purpose as possible, so other loaders that:

  - Dedicate a SCSI LUN (Logical Unit Number) to the loader mechanism,

  - Dedicate one or more LUNs to the tape transport(s)

will probably work. However, be aware that loaders with multiple access arms (the thingies that actually move the tape from a slot to a tape transport) may have problems, as the code will only try to use the first access arm. Also, the code assumes that there's only one tape transport, so loaders that have multiple transports won't work on this as written. If anyone out there wants to try to rectify this, let me know. In addition, since this code returns the device name it cobbled together from the program name, loaders that make their transports available as seperate devices won't be able to use this code, either. If you want to modify it, let me know, which brings up the issue of...

o Support? Well, I'll be here if someone wants to donate a patch/enhancement to be integrated, but I can't do too much in the way of hand-holding. You're pretty much on your own, gang...

Good Luck!

Ed Bailey
Technology Planning & Management Corp.

Internet: bailey@niehs.nih.gov
Voice: (919)361-5444, extension 419
FAX: (919)361-9680

The version of the loader control program which is included in the TapeUtils package, has been enhanced with some special features that are described in the following sections.

5.1 Usage

As explained in the **README** file above, you execute the loader by means of symbolic links named *device*-`loader` (i.e. `nrmt0h-loader`) pointing to the real `loader` executable.

This utility has a number of parameters and options. This is the help which gets displayed if you invoke it without any options:

```
# nrmt0h-loader
```

```
usage: nrmt0h-loader [-slot|-info|-reset|-eject|-status] [slot-id]

 -slot slot-id   Loads the tape stored in "slot-id" into the tape drive
                 ("slot-id" may be any loader-specific ID from -status, or
                 one of the words first, last, next, prev, and current.)

 -info           Writes to stdout the current slot-id, the number of slots,
                 and a one if the loader can go backwards, a zero if it cannot
                 (ie, a gravity stacker)

 -reset          Resets the loader to a known state - the tape in the first
                 slot is loaded

 -eject          Ejects the currently loaded tape

 -status         Writes to stdout a summary of the contents of the loader
                 and the tape drive

All commands (except -status) write the current slot-id to stdout.  With the
exception of -info, this is followed by  the tape drive's device filename, or
an error message indicating the reason the command could not be completed.

Return Codes:
0-Success, 1-Non-fatal Error (Trying to load an empty slot, etc.), 2-Fatal Error
```

5.2 New features

The enhanced version of the loader control program, which is distributed with the TapeUtils package, is fully compatible with the original version, but it includes some extra features that are not available in the original Bailey's tool.

The main new features are:

- Compatibility with the `allocate/deallocate` tools.

  The loader control program has been modified to avoid people loading or unloading cartridges if the device is allocated.

  The correct sequence to use the two tools in combination is:

  ```
  # allocate nrmt0h
  # nrmt0h-loader -slot 0          (or any other loading command)

  (Do your work safely)

  # nrmt0h-loader -eject
  # deallocate nrmt0h
  ```

  Remember to deallocate the device from the same process which allocated it.

- Support for by-user authorization.

  This feature was added to manage a very peculiar juke-box organization in use at INFN Trieste. A juke-box with a 7-slots loader is installed there. Four slots (number

3 to 6) are permanently allocated for system management tasks: we do not want people to be able to access those slots, while we want them to be able to take advantage of the remaining ones.

An authorization system was added to the loader control program, to make the administrator able to allow/deny access to some slots of the juke-box on a by-user basis.

This is done by means of a permissions file listing which users can access which slots.

The control system is rather simple. However, it covers what are supposed to be the most common situations. The main limitations are the following:

- You may enable an user to access the first n slots of the loader. You may not enable him/her to use slots x and y, neither slots x to y (if x is not the first slot).
- Restrictions are by-user, not by-group. This means that if you want to restrict access to a group of users, you must list them all. Fortunately enough, a wildcard "*" exists, that includes all the users (which is the most common case).

5.3 Loader authorization files

To enable the access control system, you need to create an authorization file for each device you mean to control. These files must be located in the directory which has been assigned to the LOADER_AUTH_FILES_PATH configuration parameter during the installation, and must be readable by any user. They must be named in the same way as the device you mean to control (i.e. `/usr/common/allocate/auth_files/nrmt0h`). They should consist of a list of entries which are couples of:

username *maximum-allowed-slot*

The *username* field is the real userid, or "*" to indicate all the users.

The *maximum-allowed-slot* parameter indicates how many slots the user is allowed to use (i.e., if the slots start from 0, setting this field to 2 will allow accessing slots 0 and 1). Use "*" to indicate all available slots.

Use "#" to begin comments.

For instance:

```
#
# Users can access only the first two slots
#
* 2
#
# Privileged users can access all the slots
#
root     *
backup   *
strizzol *
```

Users will be able to see only the slots they are allowed to access. No way for them to get info about the other slots.

If you do not need the authorization system, just do not create the authorization files.

6 Tools

6.1 TUbackup

TUbackup is a simple script which has been included in the kit (in the `tools` directory) both as an example of TapeUtils commands usage, and as a basis to build more complex tools.

This script provides an interface to perform backups using a tape loader (juke-box). It has been thought mainly as a tool for system administrator's backup tasks, to be submitted (through `crontabs`) for periodic and automatic execution.

TUbackup script is not automatically installed when you perform TapeUtils installation. If you need it, you must install it by hand.

The first part of the script includes some parameters setting that you should modify as needed. In particular, you must set up the name of the tape loader device (just something like `nrmt0h`, not `/dev/nrmt0h`), the number of slots available on the loader, and the location of TapeUtils executables.

Then, you must create a number of files containing the lists of filesystems to be backed up. Each file you build must include all the filesystems which must be backed up *on the same cartridge*, one after the other. The filesystems names must be listed on the same line, that is, each file must containg just one line, for instance:

```
/disk1 /disk2 /disk3
```

After this step, the script may be invoked with the following syntax:

`# TUbackup` *backup-level slot list-of-disks*

The *backup-level* parameter has the same meaning as for the `vdump` command: use 0 for total backup, greater numbers for incremental ones. See `vdump` command documentation for more information.

The *slot* parameter sets the loader slot on which the dump must be performed.

For instance:

```
# TUbackup 0 2 mydisks.dat
```

This would perform a total backup of the filesystems listed into `mydisks.dat` on the cartridge loaded in slot number 2.

7 Acknowledgements

The author thanks Ed Bailey of Technology Planning & Management Corp. for both writing the original loader control program, which works really great, and for exchanging opinions about the new features.

The tools described in this document have been developed and tested in a Digital Unix environment (version 4.0a). They are provided "as is". No responsibility is assumed in case of errors, bugs or unwanted side effects. Comments, bug reports and notifications about installations on other systems are welcome.

The original tape loader control program was written by Ed Bailey. For this software, rights go to him. The original kit is currently available at INFN - Sezione di Trieste, see above for pointers.

The author can be contacted at the following e-mail address: claudio.strizzolo@-trieste.infn.it.