DLTtape™ TAPE UNIT INSTALLATION

DLTtape Tape Drive Configuration on Linux

Overview

Instructions provided here are for configuring and using DLTtape drives under Linux. At Benchmark, DLTtape drives have been tested under several Linux distributions - including Slackware, Debian, RedHat and Caldera. We have tried several kernel versions ranging from 2.0.36 through 2.2.10 on Intel platforms of varying horsepower. The guidelines provided here should apply - with only minor changes required, if any - to your favorite Linux distribution.

You need a SCSI adapter on the host

If there are existing SCSI disk drives, SCSI CD-ROM or SCSI tape drives configured on your host, then you already have a SCSI adapter (or built-in interface). After shutting down Linux and powering the machine off, cable the DLTtape drive to the SCSI adapter. Select a distinct ID for the DLTtape drive on the SCSI bus. Then power up the drive and place a suitable cartridge in it.
Most of the popular SCSI adapters provide a configuration utility that can be invoked during system powerup and BIOS scan. Such utilities are handy for verifying that the cabling is good, the host adapter sees the tape drive, and there are no bus ID conflicts.

You need the Linux driver for your SCSI adapter

If you've been using other devices on your SCSI adapter under Linux, you have this driver already. If you've just installed a new SCSI card to host the DLTtape drive, you may need to build a Linux kernel that includes the appropriate driver for your adapter. Your Linux distribution should provide documentation on how to build and install a new kernel. Reboot from your kernel and make sure that the SCSI adapter driver sees your DLTtape drive. **dmesg** is useful for examining kernel startup messages.

You need the st driver

st is the generic SCSI tape driver for Linux. If you have been using other SCSI tape drives on your Linux system, your kernel already has it. If **st** is in your kernel, you should see a line similar to the following during Linux bootup for each SCSI tape drive on the system.

```
Detected scsi tape st0 at scsi1, channel 0, id 0, lun 0
Detected scsi tape st1 at scsi3, channel 0, id 3, lun 0
```

If you don't see this, you'll have to rebuild the kernel with **st** support included. Again, the documentation that came with your Linux distribution should be a good starting point for this task. After you install the kernel-source package, make sure to check the **README** files in **/usr/src/linux** for detailed version-specific rebuild requirements. If you're going to be rolling your own kernel, read the section below on **Tuning for Performance** first. This is a good time to customize your settings in the **st_options.h** file and hard-wire them into the Linux kernel. This will save you the bother of passing boot-time options to the **st** driver.

You need device-special files in /dev

Linux provides two device files for each SCSI tape drive in the system: **/dev/sti** and **/dev/nsti**, where i is a number reference to a particular drive. Drives are numbered from zero on up. Numbering is based primarily on SCSI bus number (if you have several SCSI adapters) and secondarily on SCSI ID within a bus.
A file-close on **/dev/sti** causes the tape drive to automatically rewind the cartridge. For example, after a **tar cf** on **/dev/st0**, the drive head will be positioned at beginning of tape. A subsequent **tar cf** on the same

device will overwrite the previous archive.
/dev/nsti is the no-rewind device. It can be used to create several archives, one after the other, on a single DLTtape cartridge. You can then select a particular archive to restore using the **fsf** and **bsf** operations provided by the **mt** command. This will let you locate a particular archive by seeking to the correct filemark using high-speed locate.
Check if you already have the device-special files:

```
# ls -l /dev/*st?
crw-rw---- 1 root tape 9, 128 Sep 22 15:33 /dev/nst0
crw-rw---- 1 root tape 9, 129 Sep 22 15:33 /dev/nst1
crw-rw---- 1 root tape 9, 0 Sep 22 15:33 /dev/st0
crw-rw---- 1 root tape 9, 1 Sep 22 15:33 /dev/st1
```

If you don’t see these files on your system, you’ll have to build them using the **MAKEDEV** script, which is also usually found in **/dev** and can be set up using the following command construct:

```
# ./MAKEDEV st
```

This will typically make the device-special files for up to eight SCSI tape drives, **st0** through **st7**.
On some Linux distributions, you have to run **MAKEDEV** once for each drive, as in the following example:

```
# ./MAKEDEV st0
```

You need the mt utility

On Linux, **mt** is an indispensable utility when working with DLTtape drives.
If you don't have it already on your system, install the **mt** package from your Linux distribution. Currently, there are at least two flavors of **mt** available for Linux. There is an **mt-st** package that ships with many distributions. GNU **mt** is another flavor that ships with Debian; there are probably others.
GNU **mt** responds to a version inquiry in the following:

```
# mt --version
GNU mt version 2.4.2
```

For **mt-st**, the version inquiry has the following response:

```
# mt -v
mt-st v. 0.5b
```

We find that either **mt** flavor is adequate for performing basic operations on DLTtape drives. There are minor syntax differences, however. The first thing you should do after installing **mt** is to read the manpage. It provides a vast assortment of useful commands that are too numerous to list here.

Testing your DLTtape drive

If you’ve come this far, it’s time to test that your newly-installed DLT tape drive responds properly to some basic commands.
Perform an inquiry on the drive:

```
# mt -f /dev/st0 status
drive type = Generic SCSI-2 tape
drive status = 1090519040
sense key error = 0
residue count = 0
file number = 0
block number = 0
Tape block size 0 bytes. Density code 0x41 (unknown).
Soft error count since last status=0
General status bits on (41010000):
BOT ONLINE IM_REP_EN
```

You won’t see the BOT keyword in the last line unless there’s a cartridge in the drive. Take care of that and try again.
Tar a file to the cartridge, and attempt to restore it:

```
# cd /
# tar cvf /dev/st0 tmp
# tar xvf /dev/st0 tmp
```

Until you're sure your DLTtape drive is operating correctly, use test files that you can afford to lose when you overwrite them with a restore.

Turn hardware compression off:

DLTtape drives usually have hardware compression turned on by default upon power-up. Some datasets that you want to back up may already be highly compressed via software. When backing up data that is not further compressible, there is no benefit from using the hardware compression on the DLTtape drive. You can toggle hardware compression using the **mt** command.

Using GNU **mt**,

```
# mt -f /dev/st0 datcompression 0 # turns compression off
Compression off.
# mt -f /dev/st0 datcompression 2 # turns compression on
Compression on.
# mt -f /dev/st0 datcompression 1 # reports whether on or off
Compression on.
```

Using **mt-st**,

```
# mt -f /dev/st0 compression 0 # turns compression off
# mt -f /dev/st0 compression 1 # turns compression on
```

Tuning for performance

DLTtape drives perform best when large blocksizes are used.

Check that your drive is set to variable blocksize mode, which is the default, and also the preferred mode under Unix. **mt** status should report the drive blocksize as 0 bytes. If not, this can be changed using the **mtsetblk** operation.

Specify a large per-drive buffer – say 128 Kbytes - to the **st** driver. The fastest way to do this is to pass the kernel the boot-time option **st=128**. Just specify **append="st=128"** for your kernel of choice in **/etc/lilo.conf**, run **lilo**, and reboot. Any time you rebuild the kernel from sources, you can also hard-code the buffer-size by editing the **st_options.h** file (it's in **/usr/src/linux/drivers/scsi**). Look for the **ST_BUFFER_BLOCKS** parameter in this file, and change its value to 128 (the default value is 32 Kbytes).

Explicitly specify a blocksize of 64 Kbyte or 128 Kbyte to backup and restore utilities such as **tar**:

```
# cd /
# tar cvbf 128 /dev/st0 home
# tar xvbf 128 /dev/st0 home
```

For good backup speed, you also need to make sure your disk drives and filesystems are cooperating by delivering the data fast enough. If you have several slow disk drives, experiment with software packages such as **raidtools** to see if disk striping helps. Keeping the filesystem defragmented is always a good idea; it should improve sequential file-read performance. Check your favorite Linux mirror-site for alternative backup tools. Some of them implement double-buffering, which could help reduce the frequency of repositioning on DLTtape drives.