HACKING THE TiVo

Chapter 5

Backing Up and Restoring TiVo Data

Your TiVo is both a computer system and a home entertainment appliance, and should therefore be backed up at various times. Like any other computer system, you should back up your TiVo occasionally simply to guarantee that you don't lose critical data. In the case of the TiVo, settings for features like WishLists and Season Passes are personalized aspects of the TiVo system software that you may want to protect against hardware failures or accidental loss during a TiVo software upgrade—while this has never happened to me, others have reported this problem over the years, and it's always better to be safe than sorry.

The fact that your TiVo is a home entertainment appliance that stores recordings that you may treasure is another motivation for backing up your TiVo. You may want to back up your TiVo occasionally to protect recordings that are important to you and that you want to continue to view on your TiVo. Saving such "critical" recordings to long-term media like video tape (explained later in this chapter) or to other computer systems via the network are easy and safe ways of moving or cloning data from your TiVo onto external storage where it is safe from the vagaries of hardware failures and accidental deletion.

PHYSICIAN, UN-UPGRADE THYSELF

Like all computer systems, your TiVo is ruled by its operating system software, which presents another significant reason for backing up your TiVo, although it might be frowned upon by the head TiVo office—protecting your TiVo against software upgrades, that is. As described earlier, different releases of the TiVo software have different capabilities and different levels of TiVo-induced security measures (for example, hashing). Each software release from TiVo Inc. provides enhancements such as improved recording quality, increased capabilities in WishLists, the user interface in general, and so on. However, if you've hacked your TiVo and are happy with it as it is, you may not want to get any other TiVo software release. The ability for TiVo Inc. to remotely upgrade your system can be a good thing if you just want the latest and greatest software, but it's a bad thing if you don't want your system upgraded. You can't apply the mantra, "don't fix it if it isn't broken" if someone else controls the upgrade process. To protect yourself in this case, it's important to have a backup of your TiVo so that you can roll back to the software version that you liked after being upgraded against your will, and it is also useful to discourage your TiVo from upgrading itself by setting special boot parameters. This process is described later in this chapter in "Changing TiVo Operating System Versions Using Backups."

Video extraction, which is the term used to describe the process of transferring saved video from your TiVo to another system, is an especially touchy subject in DVR circles. Video extraction and depriving TiVo Inc. of revenues from the TiVo service are the two forbidden topics on public TiVo discussion areas such as the AVS forums, but for two totally different reasons.

Depriving TiVo Inc. of service revenue (generally known as "stealing service") is wrong for obvious reasons. First, $12.95 a month or $300 for lifetime service simply isn't worth stealing. The

folks at TiVo have worked long and hard to make the TiVo the great entertainment appliance it is. Perhaps I'm old-fashioned, but I think that people deserve to be rewarded for good work. If the TiVo service is available in your area and you don't want to pay for it, that's really too bad. You should. Things are somewhat different if you live outside the US in an area where the TiVo service is unavailable, but you still want to own and use a TiVo. This book can't help you in that circumstance because some greedy fool would use those techniques in areas where the service is available, and that would be wrong. However, if you poke around on the net long enough, you can find posts and pages from people in distant areas like Australia that do not have access to the TiVo service but have worked around the problem.

Extracting video from your TiVo so that you can save it to long-term storage that you can watch using devices other than the TiVo is a totally different matter. It isn't discussed in this book because doing so may encourage the legal locusts from the MPAA and the RIAA to descend on TiVo Inc. (or me) in a frenzy of self-serving avarice, suing TiVo into a stupor. This worked with ReplayTV, and I'd hate to see TiVo get the same sort of legal cancer. Recording broadcast television and movies for resale is wrong. Recording them for your own personal use should be totally legal. They broadcast it, we received it, and we probably sat through the commercials at least once—I think we've paid our dues and fulfilled the contract. However, it's still a touchy subject that is best avoided until a more enlightened future time. Luckily, as described in the last section of this chapter, there are easy and totally legal ways of recording broadcast television that have nothing to do with your TiVo. These aspects of the topic are fair game.

Overview

There are two basic ways to back up any computer system:

- Unstructured backups, where your backup is essentially an exact copy of a disk or disk partition. This type of backup requires no information about the organization of the data on disk because you are simply making a raw, image copy of a disk or partition.
- Structured backups, where your backup is a copy of the data and relevant data structures on the disk. This type of backup must be done using a program that understands how the data on a disk or disk partition is organized, and then uses that information to traverse the disk or partition in a structured fashion.

Each of these backups techniques has specific advantages. Unstructured backups are easy to do, require no special software, and serve as a complete snapshot of an entire disk or partition at a given moment in time, down to files that the system has marked for deletion but which have not yet been purged from or overwritten on the disk. Unstructured backups of the TiVo use the Linux version of one of the original Unix utilities, an age-old utility called *dd*, which stands for "dump data." The dd program does exactly that by leveraging the fact that all native system data on Unix/Linux systems is stored as a stream of bytes and can be accessed in that way. As explained in more detail later in this chapter, the dd utility has two mandatory arguments, which are the names of its input and output files.

Structured backups take less space than unstructured backups because they are performed by utilities that understand the organization of the files, directories, and relevant data structures on the filesystems they back up. They access the data on the disk or partition that you are backing up through actual filesystem data structures, and therefore only back up the portions of the filesystem that actually contain data. If you only have 1 GB of data on a 10 MB partition, structured backups will only require 1 GB of storage space (at most, and perhaps less, if you are doing compressed backups).

Several structured backup utilities are available for the TiVo, the best being MFS Tools utilities. The **mfstool** backup utility is specially designed to back up all of the types of filesystems that the TiVo uses, and features many different options to customize what you are backing up and how you are backing it up. Doing structured backups of your TiVo using the **mfstool** backup command is discussed later in this chapter, in the section "Creating Backups Using MFS Tools." The structured backups of a TiVo drive that do not include your recordings are usually between 100 and 200 MB in size.

The MFS Tools utilities recognize the format of all of the types of filesystems used by your TiVo, from standard Linux **ext2** filesystems to the MFS filesystems that are specific to the TiVo, and back each up differently. On a hacked TiVo, you could also choose to back up the data in the TiVo's standard Linux **ext2** filesystems (**hda4**, **hda7**, and **hda9**) by using the standard Linux **tar** (tape archiver) and **cpio** (Copy Input to Output) utilities. Each uses its own archive formats, but they are quite common in Linux/Unix circles. The primary difference between the two utilities is that the **tar** utility cannot back up special Linux/Unix files such as the entries in the **/dev** and **/proc** directories, but otherwise is the ubiquitous standard for archiving Linux/Unix directories. The **tar** utility takes the name of one or more directories or files as a command-line argument, and creates an **ar** archive file that contains the specified directories and files. The **cpio** utility is much less widely used, but has more sophisticated capabilities. It can back up any type of Linux/Unix file, but requires that a list of files and directories to back up be supplied as its standard input, and writes an archive of those files and directories to its standard output (which users typically redirect into a file).

NOTE

While the Linux/Unix structured backup utilities discussed earlier can be useful for backing up selected portions of the Linux filesystems on your TiVo, using the MFS Tools utilities is the only way to do a structured backup of your TiVo that you can subsequently use to restore your TiVo to a completely usable state.

The downside of structured backups is that they are very sensitive to the consistency and condition of the filesystem. If the filesystem is corrupted, the utility that you are using to make structured backups may not be able to follow the organization of files and directories on the

disk, and may simply bail out. On TiVo filesystems that store information in a standard Linux filesystem format, such as the root and /var partitions' use of the ext2 filesystem format, you can usually correct this sort of problem by running the e2fsck utility to clean up those filesystems before doing the backup. This won't help if there are problems with any of your MFS filesystems. Similarly, if the underlying hardware on which a filesystem is stored begins to go bad, you may not be able to make structured backups due to read errors on the hard drive.

IF TIVOS CRASH AND STRUCTURED BACKUPS FAIL...

If one of the drives in your TiVo is going bad and occasionally flakes out (i.e., your TiVo crashes or structured backups fail), creating an unstructured backup of that disk may be the solution to your problem. You can use the **dd** program to create an image backup of the entire disk, and then either replace the disk or reformat it. (The former is preferable, but the latter may work.) After you have replaced or reformatted the disk, you can then restore the backup to the new disk (as explained later in this chapter), and then retry the structured backup. If you are lucky and the reason that the backup failed and your TiVo crashed was repeated read/write errors to the disk, you just might be able to recover all of your TiVo settings and existing recordings.

When to Back Up?

To some extent, the differences between the two types of backups dictate when it is appropriate to use one over the other. You can make an image backup at any time, because it always uses the same amount of space as the media that you are backing up. In other words, making an image copy of a 30 GB disk will always require 30 GB of backup storage, regardless of whether the disk contains 1 byte of data or 29 GB of data. If you can afford the disk space, keeping an image backup around is never a bad idea. However, since unstructured backups save an exact snapshot of the entire disk, people rarely do them except when they first get a TiVo, and perhaps later when the TiVo disk is almost full of recordings that they want to save.

TIP

Because unstructured backup images are so large, they are often split into multiple files, which are therefore small enough to fit on removable media (such as CDs or backup tapes for offline storage) until you need them. Use the Linux/Unix **split** command and the Linux/Windows RAR utility for this purpose.

If conserving disk space or minimizing the amount of potentially extraneous information in your backups is a concern, the best time to use the MFS Tools to do a structured backup of your TiVo is the day that you get it, before going through the Guided Setup process. This gives you a completely virginal image of your TiVo's hard drive. You can then restore this to your TiVo at any time to start over from ground zero. A virginal backup image can also come in handy if you or a friend encounters hardware problems and needs to completely restore a TiVo to its original, out-of-the-box state.

For personal reuse, you may find it preferable to use the MFS Tools to do a structured backup of your TiVo immediately after going through the Guided Setup process so that the backup contains the information about your local TiVo dialup connection. Disk space is cheap nowadays. Similarly, it is also a good idea to do a backup after any new TiVo software release in order to eliminate the need to re-upgrade your system if you have to reinstall from backups for one reason or another.

One thing to remember about backing up your TiVo using the MFS Tools utilities is that, unlike many structured backup utilities, the MFS Tools do not perform incremental backups. Every backup of a primary TiVo disk done with the MFS Tools is a complete backup of the subset of the partitions on that TiVo disk (and the data that they contain), which is required to successfully boot your TiVo. This means that MFS Tools backups are always relatively large even if little has changed on your TiVo since the last time you did an MFS Tools backup. This also means that restoring one MFS Tools backup to a disk on which a previous MFS Tools backup has been restored will completely overwrite the previous backup data.

Finding Room for Backups

Whether you're creating structured or unstructured backups, the fact remains that backup files take up a fair amount of space. Unstructured backups of disks and partitions require exactly the same amount of space as each disk or partition that you back up. Structured backups of a TiVo disk, even without saving any of your recordings, generally take between 200 and 800 MB. Where should you put your backups?

Disk space is cheap nowadays, but getting to it may be an issue. The primary hard drives in most people's computer systems are relatively large, usually ranging from 10 GB to well over 100 GB. TiVo tools CDs, such as BATBD, use RAM disks to hold their filesystems and are primarily designed to give you access to the TiVo disks. Luckily, there's an easy solution to this problem, which capitalizes on the flexibility of Linux and its ability to interoperate with the different types of filesystems used on a variety of personal computers.

NOTE

The versions of Linux provided on the BATBD CD can only access storage located on Windows disks that have been formatted with the FAT32 filesystem. If the filesystems on your personal computer are formatted using the NTFS (NT File System) format, you will not be able to access them after booting from the BATBD CD. Chalk up another piece of proprietary crap to Microsoft.

If your primary desktop machine is a Windows system and you don't know what type of filesystem it uses, you can find out from Windows by opening the My Computer folder on your desktop, right-clicking the icon for your system's C drive, and then selecting Properties from the context-sensitive menu that displays. The resulting dialog box displays information about the disk including the type of filesystem that it contains, as shown in Figure 5.1.

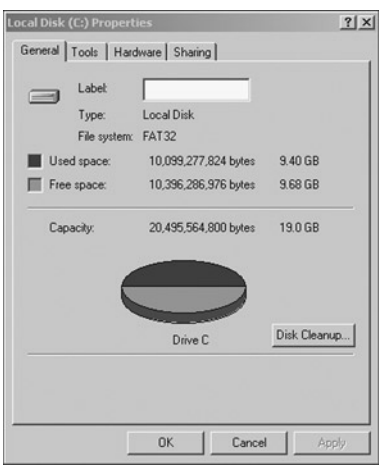


FIGURE 5.1 *Determining the type of filesystem on a Windows drive*

As discussed earlier, even the kernels on TiVo tools CDs that do byte-swapping only activate byte-swapping for the devices attached as the slave device on your primary IDE interface (**/dev/hdb**, in Linux terms), the master drive attached to your secondary IDE interface (**/dev/hdc**, in Linux terms), and the slave drive attached to your secondary IDE interface (**/dev/hdd**, in Linux terms). The master drive on your primary IDE interface is never byte-swapped, and therefore provides access to the native filesystems on your personal computer. All you have to do is make them available to the TiVo tools CD from which you've booted. This can be done with a single Linux command.

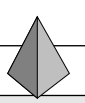
TIP

When doing backups, you do not need to boot with the byte-swapped kernel even if you are using a Series 1 system. When you restore the backup, the MFS Tools utilities' **mfstool restore** command provides options that make it easy for you to restore both byte-swapped and "normal" backups—as discussed in Chapter 4, the MFS Tools utilities provide internal support for byte-swapping when necessary, regardless of what the kernel provides. However, if you are using a Series 1 system, you always want to boot using a byte-swapped kernel when you are restoring a backup to a TiVo disk. This will enable you to use the **pdisk** command to verify that the TiVo disk has been restored correctly. Mounting and unmounting disks and moving them from one system to another is tedious enough without doing it multiple times because the restore used the wrong byte order.

When you boot from a TiVo tools CD, such as BATBD, you are running Linux. Linux uses its **mount** command to add existing disks and partitions to the filesystem. The syntax of the **mount** command is as follows:

```
mount -t <type> <partition> <mountpoint>
```

In this example, <type> is the type of partition that you are mounting, which would be one of **vfat** (the Linux name for Windows FAT32 partitions), or **ext2** if your desktop system is a Linux system that uses the default **ext2** partition type. The <partition> argument is the Linux name for the disk partition that you want to mount, probably **/dev/hda1** for a Windows C drive and any of **/dev/hda1** through **/dev/hda8** for a Linux system (depending on how you partitioned your disk when you installed Linux). The <mountpoint> argument is the name of a directory in the Linux filesystem (the RAM disk, if you've booted from BATBD) through which you want to be able to access the partition that you're mounting. Linux traditionally puts directories on whichever external filesystem is to be mounted (known as *mountpoints*) in the directory **/mnt**.



TIP

You can use the Linux **fdisk** command to quickly list the types of filesystems on a Linux disk. For example, the command `fdisk -l /dev/hda` would produce a list of all of the partitions on the drive /dev/hda and their types.

If your desktop computer is a Windows system where the C drive is a FAT32 partition and you have booted from the BATBD CD, you could mount your Windows disk on the directory **/mnt/windows** using the following command:

```
mount -t vfat /dev/hda1 /mnt/windows
```

After executing this command, you could create TiVo backup files in the directory **/mnt/windows**, which you would then find in the C: directory on your Windows system after booting it from its own disk rather than the BATBD CD.

If your desktop computer is a Linux system, use the **df** command to see how its partitions are organized before you reboot your system from the BATBD disk. For example, the output of the **df** on my primary Linux system would look like the following:

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda7             10080488    987164   8581256  11% /
/dev/hda1               101089     29810     66060  32% /boot
/dev/hda8             30668576  15242724  13867952  87% /home
none                    515532         0    515532   0% /dev/shm
/dev/hda2              2016044       520   1913112   1% /tmp
/dev/hda5             20161172   4249540  14887492  23% /usr
/dev/hda6             15116836   2799492  11549440  20% /usr/local
/dev/hdb1            196015808  83683884 102374840  45% /opt2
```

In this case, the partition **/dev/hda8** is the partition where home directories are located and the partition also has around 13 GB of free disk space, which is enough for most structured TiVo backups. After booting from the BATBD CD, you could therefore mount this filesystem on the directory **/mnt/home** using the following command:

```
mount -t ext2 /dev/hda3 /mnt/home
```

In this case, the **-t ext2** argument is actually unnecessary, because **ext2** is the default partition type for the **mount** command—it is shown here for the sake of completeness. After executing this command, you could create TiVo backup files in the directory **/mnt/home**, which you could then find in the **/home** directory on your Linux system after booting it from its own kernel rather than the BATBD CD.

If you are using a Windows system and your C drive turns out to be formatted using an NTFS filesystem, all is not lost—you can always add another disk to your Windows system, boot into Windows, and format the new drive using the FAT32 filesystem. You then can shut down the system again, add your TiVo disk to this system, and mount the newly-created FAT32 partition as described earlier, using that disk as storage for your backup files.

If you have absolutely no disk space and want to back up a TiVo disk solely for the purpose of immediately restoring it to another disk, you can take advantage of the ability of Linux to connect the output of one program to the input of another. In Linux terms, this is done using a pipe, which is represented on the command-line as the "|" symbol. In this way, you can use the **mfstool backup** command to back up your existing disk, writing the backup file to what is known as *standard output*, which the corresponding **mfstool restore** command will then use as its *standard input*. This process is described in more detail later in this chapter in "Connecting Backup and Restore Commands Using a Pipe."

Creating Image Backups Using dd

As mentioned earlier, the **dd** command is one of the oldest and most primal Unix commands around. The **dd** command reads data from one source and writes it to another, with some control over the size of the chunks in which it is read and written at each end, plus the ability to do various forms of data manipulation in between. You'll only need to know about a few of these bells and whistles to use the **dd** command to make image backups of a disk or disk partition from your TiVo. However, it's always nice to know that there are other knobs to turn if future developments in TiVo-land require them.

The first step in creating an image backup of a TiVo disk or disk partition is to take the disk drive out of your TiVo and put it into your desktop PC, as explained in Chapter 3, "Exploring Your TiVo Hardware," in the section "Working with TiVo Disk Drives." As discussed in that section, make sure that you jumper the disk correctly so that it does not conflict with any existing disk in your desktop PC. Also note the IDE interface to which you attached the disk, so that you can be sure that you are backing up the right disk. Few things would be as embarrassing or irritating as trying to use a backup image to restore or expand your TiVo, only to find out that the backup image contained data from the wrong disk or partition.

NOTE

As explained in the section of Chapter 4 that discusses bootable TiVo tools disks, any of the byte-swapped kernels on these disks (necessary for mounting TiVo Series 1 disk partitions) only byte-swap the secondary disk on your primary IDE interface **(/dev/hdb**), and the primary and secondary disks on your secondary IDE interfaces (/**dev/hdc and /dev/hdd**, respectively. If you want to access the partitions on a TiVo Series 1 disk, that disk must not be attached to your PC as the master drive on your primary IDE interface.

Once the disk is successfully mounted in your PC, you can use the **dd** command to make a backup image of an entire disk at any time, regardless of which of the TiVo kernels on the BATBD disk you have booted from. Reading raw data from an entire disk does not require that you can directly access the specific partitions on the disk, only that you can access the disk itself. (Once you've booted from the BATBD disk, follow the naming conventions given in Chapter 3 for accessing TiVo disks mounted in your PC.) However, if you want to make image backups of specific partitions from a TiVo disk, you will need to boot using a kernel that understands the TiVo's partition table format, and which has the appropriate byte-order for the type of TiVo from which the disk you are copying was taken. If you are working with a TiVo Series 1 disk, you will need to use the BATBD boot option that specifies that it is designed to access TiVo partitions. Since the disks from Series 2 machines do not require byte-swapping to access their partition table, you could boot using either of the other boot options from the BATBD CD to access partitions on a TiVo Series 2 disk. These are the

Series 2 or non-swapped Series 1 boot options, though the former certainly seems to make more sense.

TIP

If you are using a desktop Linux system for your TiVo hacking purposes and want to mount TiVo Series 1 or Series 2 disks without using the BATBD disk, you will need to recompile your kernel. To do so, simply enable the Macintosh Partition Support in the filesystems section of the kernel Configuration menu. Once you've rebuilt the kernel, you can mount the **ext2** partitions from a TiVo Series 2 disk directly. You can also mount the **ext2** partitions from a TiVo Series 1 disk by passing the **hdX=swap boot** option on the kernel command line, where **X** is the letter corresponding to the TiVo disk in your system. You can also create an entry for byte-swapped booting in the configuration files used by the Linux boot loaders, GRUB or LILO, to simplify support for Series 1 disks in the future.

After you've booted your system with the appropriate kernel options, you can use the **dd** command to create an unstructured backup of a TiVo disk by using a command like the following:

```
dd if=<disk> of=<backup-file> bs=32k
```

This tells the **dd** command to read from the disk specified as <disk> (probably one of **/dev/hda**, **/dev/hdb**, **/dev/hdc**, or **/dev/hdd**) and write to the output file <backup-file>. The **bs** option tells the **dd** command to use 32K as both the input and output block sizes, which reflects the block size used on TiVo filesystems.

You can use the **dd** command to create an unstructured backup of a specific partition on your TiVo disk by using a command like the following:

```
dd if=<partition> of=<backup-file> bs=32k
```

This tells the **dd** command to read from the disk specified as <disk> (probably one of **/dev/hdXY**, where **X** is the identifier for the disk and **Y** is the partition number) and write to the output file <backup-file>. As in the previous example, the **bs** option tells the **dd** command to use 32K as both the input and output block sizes, which reflects the block size used on TiVo filesystems.

The **dd** command takes a while to run, given that it is transferring a fairly large amount of data. Unfortunately, the standard versions of the **dd** command provide no visual feedback while they execute. For people who absolutely must have a progress indicator, a specially hacked version of the **dd** command with a progress indicator is available on the net at http://tivohack.sourceforge.net/files/fileutils-4.1.bonehead.tar.gz. (I have nothing to do with the naming convention for this file, which is intended to reflect the fact that displaying the progress indicator slows down the overall performance of this version of the **dd** command by approximately 10%.) To display the progress indicator, pass the progress=1 option on the **dd** command line.

EXPLORING TIVO THROUGH VIRTUAL FILESYSTEMS

If you are using a desktop Linux system for your TiVo hacking, making image backups of partitions provides interesting hacking opportunities. Once you use **dd** to clone a disk partition to a file, you can then use Linux's loopback filesystem to mount that file as though it were a physical partition on your Linux system. This only works for partitions that are in a format that your desktop system understands, such as the **ext2** partitions in partitions 4, 7, and 9 of any primary TiVo disk.

If you want to mount partition images from a Series 1 system (which is big-endian) on a standard desktop PC (which is little-endian), you will need to convert the byte-order of the filesystem image using the **dd** command's **conv=swab** conversion option, as in the following example:

```
dd if=series1_hda9.img of=series1_hda9_pc_style.img conv=swab
```

You can then mount the "un-byteswapped" image (**series1_hda9_pc_style.img**, from the preceding example) using the **mount** commands described later in this section. For more information about "endian-ness," see the section of Chapter 4 on the **pdisk** command.

Mounting filesystem images enables you to explore the contents of the parent partition without requiring the disk that contains the filesystem still is physically attached in your desktop system. This can be handy if, for example, you would like to actually use your TiVo at the same time that you are learning more about its internals.

To mount a partition image as a file, use a command like the following:

```
mount -o loop <filename> <mountpoint>
```

In this example, <filename> is the name of the file to which you wrote the partition image (using a command like **dd**, as explained later in this chapter), and <mountpoint> is the name of the directory on which you want to mount the partition image. For example, the following command would mount the partition image in the file "hda_part4.img" on the directory **/mnt/tivo/part4**:

```
mount -o loop hda_part4.img /mnt/tivo/part4
```

After you have mounted a partition image, you can redirect the directory to it, move around in its files and directories; you can create, edit, and delete files, and generally do anything that you could do if it were an actual disk partition. This also includes running scripts on any Linux system, as well as programs—if you happen to be running a compatible version of Linux on a PPC or MIPS box. (Not much chance of that these days, but you never know...)

If you mount an image file, you must unmount it before shutting down your system or, like any other filesystem, you should use the **fsck** command to verify its integrity before reusing it.

TIP

If, for some reason, you have problems using the specified **dd** command due to errors, try executing the following **dd** command, which uses some of the **dd** command's "forgiveness" options:

```
dd if=<partition> of=<backup-file> bs=32k conv=noerror,sync
```

These options cause the **dd** command to proceed past simple errors that you may encounter if your source drive is having problems. The **sync** option synchronizes, reads, and writes—even after a read error—by padding incomplete reads with the appropriate number of NULL bytes.

Again, whether you are backing up a partition or an entire disk, the **dd** command takes a long time. Creating an image backup of a TiVo partition is a great opportunity to walk your dog around the block or to have a cigarette. Creating an image backup of an entire TiVo disk is a great opportunity to make dinner or go shopping.

Creating Backups Using MFS Tools

The unstructured backups discussed in the previous section are typically huge because they are the same size as the disk or partition that you are backing up. The MFS Tools utilities, discussed in detail in Chapter 4, create structured backups of TiVo disks by walking through the disk's partition table and the actual data structures contained in MFS filesystems. The MFS Tools utilities also conserve some space by automatically skipping certain partitions during the backup process, such as the duplicate kernel and root filesystem partitions on a bootable TiVo disk. As explained in Chapter 4, you can further reduce (or increase) the size of the backups produced by the **mfstool backup** utility by passing various command-line options.

This section explains how to perform a variety of different types of backups using the **mfstool backup** command from MFS Tools utilities. As with the previous section, you will need to have mounted the TiVo disk(s) that you want to back up in your PC before preceding; however, you do not have to boot with any special command-line options if you are using Version 2 of the MFS Tools utilities. Knowledge of the partition table and byte-swapping requirements of TiVo disks was built into Versions 2 and later of these utilities—all of which you need to know to execute the appropriate command. You may need to do a bit more when restoring the data to verify that it was written with the appropriate byte-order. (More about that later in this chapter, in the section entitled "Restoring Backups Using MFS Tools.")

NOTE

If you are unclear on how to attach a TiVo drive to your personal computer or aren't sure of the naming conventions used to access such drives after doing so, please review the section in Chapter 3, entitled "Working with TiVo Disks."

The following information explains how to use the basic capabilities of the MFS Tools utilities, beginning with simple structured backups that use the default settings compiled into the MFS Tools utilities, then moving on to more advanced types of backups. All of these backup commands use the **mfstool backup** command from MFS Tools utilities, showing how different options to the command can affect the size and contents of the backup files that you create.

Creating a Simple Backup Using MFS Tools

The simplest type of structured backup that you can do using the **mfstool backup** command requires only one command-line option, which is the name of the backup file that you want the utility to create. After booting from the BATBD disk, to back up a TiVo disk that is attached to your personal computer as the slave drive on your secondary IDE interface, you would execute a command like the following:

```
mfstool backup -o <output-file> /dev/hdd
```

In this example, <backup-file> is the name of the file to which you want to write the backup data. If you followed the conventions discussed earlier in this chapter for mounting a partition from your primary Windows or Linux disk so that you can access it from the BATBD disk, this file should be the full path of a file on the appropriate disk. For example, if your Windows C drive is mounted as **/mnt/**windows, you could specify a filename such as **/mnt/windows/my_tivo.bak**. If the home partition of a desktop Linux system is mounted as /mnt/linux, you could specify a filename such as **/mnt/home/my_tivo.bak**. Using the Windows disk as an example, you would execute a command like the following:

```
mfstool backup -o /mnt/windows/my_tivo.bak /dev/hdd
```

The **mfstool backup** command displays output like the following during the backup process:

```
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
Uncompressed backup size: 1309 megabytes
Backing up NNN of 1309 megabytes (M.MM%)
Backup done!
```

The line beginning with "Source Drive" shows the **mfstool backup** command's idea of the capacity of the drive you're backing up, which in this example is a 15MB Quantum disk from a TiVo Series 1 machine. While the **mfstool backup** command is running, the line reading

"Backing up NNN of 1309 megabytes (M.MM%)" is continually updated to show the progress of the backup command. After the command completes its task, congratulations! You've backed up your TiVo!

You can determine the actual size of the backup file produced by the **mfstool backup** command by using the Linux **ls** command, as in the following example:

```
ls /mnt/windows/my_tivo.bak
-rwxr-xr-x   1 root     root     1374030336 Jun 26 12:19 /mnt/windows/my_tivo.bak
```

In this case, the file is 1,374,030,336 bytes in size, which is close to what was predicted.

NOTE

The actual size of your backups depends on the size of your original drive, the version of the TiVo software that is installed on the disk, any additional information that you may have put in your drive's **/var** partition, and the backup options that you specify. The sizes shown in the examples throughout this chapter will almost certainly not match the size of your backups.

Since this backup command used the defaults provided by the **mfstool backup** command, it's useful to know what they are and what the backup file contains. With no options other than the **-o** option to specify the name of the output file, a backup file produced by the **mfstool backup** command contains the following:

- A copy of the partition map from the specified disk
- A copy of the bootpage from the specified disk
- A compressed copy of the active kernel partition from the specified disk
- A compressed copy of the active root filesystem partition from the specified disk
- A compressed copy of the /var partition from the specified disk
- A backup of the entries in the MFS partitions on the specified disk whose filesystem identifier (fsid) is less than 2000. These are all of the files in the MFS Application and Media regions on the disk that are required for normal operation of your TiVo.

When you restore a backup file created by the **mfstool backup** command, the **mfstool restore** command uses the copy of the original drive's partition map to create the necessary filesystems before restoring their contents. In the case of the Linux swap partition on the TiVo disk, the **mfstool restore** command simply creates an empty partition of the appropriate size (or larger, depending on **mfstool restore** options) and correctly initializes it for use as a swap partition. Backing up a swap partition would be a total waste of time because it is used only by the operating system as a temporary location to provide virtual memory support.

For more details about the restore process and the contents of a backup file created using the **mfstool backup** command, see the section "Restoring Backups Created Using MFS Tools," later in this chapter.

Creating a Compressed Backup Using MFS Tools

The previous section explained how to create a simple structured TiVo backup using the **mfstool backup** command. While still much smaller than the physical capacity of the TiVo disk that you were backing up, these backups are still fairly large—816 MB or so is nothing to sneeze at. A decade or two ago, this was beyond the maximum storage capacity of a good-sized mainframe. Even though disk space is cheap now (I paid $160 or so for my last 200 GB drive), saving disk space whenever possible is never a bad idea. A classic computer maxim goes like this: "The amount of disk space required by users always expands to fill all available storage." This certainly is even truer when programs create 816 MB files with a single command.

The **mfstool backup** command provides built-in options for compressing backup files as they are created. For the hard-core nerds among us, this is done through the used of the Linux **zlib** library, which implements a standard Lempel-Zev compression scheme to reduce disk consumption in output files. For the rest of us, "the files are smaller."

The **mfstool backup** command provides 10 levels of compression (levels 0 through 9), which reflect the tradeoff between compression and execution time. Obviously, compressing data takes some time, which means that creating a compressed backup takes longer than creating a standard, uncompressed backup file. Higher levels of compression do substantial amounts of extra processing to reduce the size of your backup files, but may not have as radical an impact on the size of your backup files as the middle compression levels do. At some point, the savings provided by higher levels of compression isn't worth the time it takes to completely minimize the size of your backup files——saving two or five megabytes of a file that is still a few hundred megabytes in size just may not be worth it.

When using the **mfstool backup** command, compression options are specified using a command-line option that identifies the compression level for which you're looking. These options are therefore -1 through -9, with -9 being the maximum compression level possible. The compression value recommended in the documentation for the **mfstool backup** command itself is -6, which is the one that I typically use. As noted in the MFS Tools documentation, this compression option seems to provide the best tradeoff between savings in backup file size and slower performance of the **mfstool backup command.**

Using a mounted Windows disk as an example of the destination for the backup file that you are creating, the command you would type to perform a compressed backup at compression level 6 is the following:

```
mfstool backup -6 -o /mnt/windows/my_tivo.bak /dev/hdd
```

The **mfstool backup** command displays output like the following during the backup process:

```
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
Uncompressed backup size: 1309 megabytes
Backing up NNN of 1309 megabytes (M.MM%)
Backup done!
```

This command displays the same output as the **mfstool backup** command without the compressed backup option. The real difference is in the final size of the output file, which you can only determine by actually examining the file by using the Linux **ls** command, as in the following example:

```
ls /mnt/windows/my_tivo.bak
-rwxr-xr-x   1 root     root     601924044 Jun 26 12:38 /mnt/windows/my_tivo.bak
```

In this case, the file is 601,924,044 bytes in size, which essentially is half of the predicted size of the uncompressed backup.

Backing Up an Entire TiVo Disk Using MFS Tools

As mentioned earlier, the default action of the **mfstool backup** command is to back up files only within the MFS partitions of a disk whose filesystem identifier is less than 2000. This seemingly arbitrary number is actually a carefully chosen value that guarantees that the backup file will contain all of the portions of the MFS filesystem(s) that are required for normal operation of your TiVo, without backing up the files that truly take up the majority of the space in any MFS filesystem—your recordings. However, there are a few cases in which you would want to back up everything, namely:

- When backing up a failing TiVo disk so that you can restore it to an identical disk in order to replace the failing disk.
- When backing up a TiVo disk so that you can subsequently restore it to a larger disk and use the additional space provided by the larger disk to increase the storage capacity of your TiVo.
- When backing up your existing TiVo disk simply to protect treasured recordings against accidental deletion or loss due to hardware failure (if the disk itself goes bad).

In any of these cases, you can use additional **mfstool backup** options to increase the amount of information that is being backed up, customize the specific types of items being backed up based on the amount of disk space they consume, or simply back up everything.

Backing up the entire contents of a TiVo is easy enough to do, requiring that you add a single command-line option. The **mfstool backup** option for backing up the entire contents of one or more TiVo drives is the **-a** (all) option. The following example shows the sample command

and resulting output of backing up the entire contents of a TiVo disk that is connected to your personal computer as **/dev/hdd** (the slave drive on your secondary IDE interface):

```
mfstool backup -6 -a -o /mnt/windows/my_tivo.bak /dev/hdd
```

The **mfstool backup** command displays output like the following during the backup process:

```
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
Uncompressed backup size: 1309 megabytes
Backing up NNN of 1309 megabytes (M.MM%)
Backup done!
```

Although in this case, the size of the backup is the same as our previous backup; full-disk backups are typically much larger. This is not surprising because you are backing up everything contained in the entire disk. As long as you can afford the backup space required to save a backup file of this size, it is nice to have the ability to back up all of your favorite recording at one time.

TIP

Before doing a complete backup that you plan to keep around for a while, take the time to purge any non-essential recordings from the TiVo. You should also use the **backdoor** commands discussed in Chapter 2, "TiVo Tips and Tricks," in the section "Using Thumb-Thumb-Thumb Codes" to display the listings for any Teleworld or other promotional recordings that may still be stored on your disk and delete them. No point in backing up anything that you don't really want to save!

Backing Up Multiple-Disk TiVo Systems

As mentioned when discussing the BlessTiVo command in Chapter 4, "The Hacker's Tool Chest of TiVo Tools," (and discussed in more detail in Chapter 6, "Expanding Your TiVo's Storage Capacity"), adding a second drive to your TiVo effectively marries the two disks so that they "always" must be used together. What this really means is that you cannot remove the second drive and continue to use the TiVo with just its primary disk. Since the TiVo expects to access and use storage on the second drive, it won't take long for the TiVo to attempt to access the second drive and (of course) fail. In the computer biz, this is known as "a bad thing." Fortunately, you can effectively divorce two drives by backing them up together, and then restoring the combined backup to a larger, single drive. The **mfstool backup** command stores a copy of the partition map for both the TiVo drives, but it does not require that the restore be performed on two drives—when restoring a dual-drive backup to a single TiVo drive with sufficient space, the **mfstool restore** command simply re-creates a sufficient number of partitions to hold the restored data on the new, larger drive.

Backing up a dual-drive TiVo is done exactly like backing up a single-drive TiVo, with two exceptions:

- You must supply the **-s** (shrink) command-line option to shrink the set of volumes being backed up as much as possible. This causes the **mfstool backup** command to focus on the data contained in any MFS partitions, rather than trying to preserve the actual structure of the volumes and partitions on your disks.
- You must supply the names of both TiVo drives on the TiVo command-line and in the correct order, of course.

A sample command-line for backing up a dual-drive TiVo, whose disks are found at **/dev/hdb** and **/dev/hdd** (in other words, as slave drives on both of your IDE interfaces) with output from the **mfs backup** command, would look something like the following:

```
mfstool backup -6 -s -o /mnt/windows/my_tivo.bak /dev/hdb /dev/hdd
```

The **mfstool backup** command displays output like the following during the backup process:

```
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
      - Upgraded to 57 hours
Uncompressed backup size: 1309 megabytes
Backing up NNN of 1309 megabytes (M.MM%)
Backup done!
```

This command displays the same output as any other **mfstool backup** command, with the exception of the information provided about the size of both of the TiVo drives. As with the previous example, the relationship between the estimated, uncompressed size of the backup and the actual size of the backup file is only a guess—you'll need to use the **ls** command to see the actual resulting size of the output file.

After you've created a single backup file that combines information about both drives of a dual-drive TiVo system, you can restore it to a single drive with sufficient available storage, or to a larger set of drives if you want to maximize your storage at the same time. Restoring backups is explained later in this chapter; restoring a single backup file to a pair of disks is explained in Chapter 6, "Expanding Your TiVo's Storage Capacity," as part of the general discussion of how to add a second drive to your system.

Advanced Backup Options

As discussed in Chapter 3, the options available to the **mfstool backup** command provide a few options that enable you to modify the contents of a structured backup file in general and specific ways. This section highlights these options to help you think about ways of customizing your backups. For the most part, I tend to stick with "standard" backups, rarely taking advantage of the more advanced **mfstool backup** options with the exception of the **-v** option (to exclude **/var** from the backups). With other advanced options, it's tricky to know exactly

what a specific backup contains (or does not contain) without actually restoring it and seeing what you get. However, your mileage may vary, so here we go.

The discussion of the TiVo's Media File System (MFS) in Chapter 9 presents ways of identifying the MFS entities associated with specific recordings. If you want to want to create a TiVo backup that is guaranteed to include a specific recording, you can use the **mfstool backup** command's **-f** <fsid> option to ensure that the backup includes the specified recording. This option actually tells the **mfstool backup** command to include all videos with **fsids** less than or equal to the specified <fsid>, which probably means that you'll get other recordings than the one in which you're specifically interested. However, this also means that you won't get other recordings <fsid>s that are greater than the one that in which you're interested.

The **-f** <fsid> option provides a good compromise between minimizing the size of your backup while ensuring that your backup includes a specific recording, without creating a backup of every recording on your TiVo. Continuing with the examples used in the previous sections, the following command would create a compressed backup file containing all video streams with <fsid>s that are less than 2877, which we'll assume is one greater than the <fsid> of the recording we're interested in preserving:

```
mfstool backup -6 -f 2877 -o /mnt/windows/my_tivo.bak /dev/hdb
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
Uncompressed backup size: XXXX megabytes
Backing up NNN of XXXX megabytes (M.MM%)
Backup done!
```

An alternative to just backing up streams with <fsid>s that are less than a specified number (or the default value of 2000) is to use the **mfstool backup** command's **-l** <size> option to limit the backup to only streams that are under a certain size. This size is based on the actual amount of data that the files contain, rather than the amount of space allocated to each file (which would differ based on the allocation units used for that TiVo filesystem). The assumption behind this option is that files larger than the specified limit will be recordings (which you would not want to save) because the native animations used by the TiVo are all relatively small. Safe values for this <size> cutoff are 32 or 64 MB. This option cannot be used with the **-f** option, because there may be files with <fsids> that are less than the specified value, but which is larger than the specified <size> cutoff.

Continuing with the examples used in the previous sections, the following command would create a compressed backup file containing all video streams with <size> less that 32 MB:

```
mfstool backup -6 -l 32 -o /mnt/windows/my_tivo.bak /dev/hdb
```

The **mfstool backup** command displays output like the following during the backup process:

```
Scanning source drive.  Please wait a moment.
Source drive size is 15 hours
```

```
Uncompressed backup size: XXXX megabytes
Backing up NNN of XXXX megabytes (M.MM%)
Backup done!
```

If you would rather base the size cutoff on the actual amount of space that is actually allocated for a file, rather than the amount of space that is actually used, you can use the **-t** <size> option instead of the **-l** <size> option. In a final permutation between allocated and actual size, you can use the **-T** <size> option to back up all of the space allocated to files that actually use <size> or less of that space.

Backing Up Selected Information from Your TiVo

The previous sections discussed how to do backups of your TiVo, which then can be restored to create a new bootable TiVo disk or disk set. While the **mfstool backup** command provides options for reducing (or simply controlling) the items that are included in the backup files that it produces, it does not support incremental backups. *Incremental backups* is the term used for backups that only contain things that have changed since the previous backup was done. The combination of these two facts has the following implications for backups created using the **mfstool backup** command:

- They are always large.
- Each backup contains mostly the same files as the previous backup.
- Each backup contains identical copies of system files that probably have not changed since the previous backup.

If you aren't interested in backing up your recordings and already have a backup of your TiVo system software in a safe location, there are still several things that you have probably spent a fair amount of time crafting, and therefore would be distressed to lose. These are features like Season Passes and WishLists that you've built up over time. Luckily, a thoughtful TiVo hacker named "angra" was kind enough to create backup and restore commands. If you run these commands on your TiVo, it will save and restore these items to and from text files that you can easily preserve for safekeeping by copying them to another machine. These scripts are located in the BATBD CD's settings-backup directory.

NOTE

The settings **backup** and **restore** commands must be executed on your TiVo, unlike the **dd** and MFS Tool utilities. Backing up and restoring your TiVo settings therefore requires that you already have configured your TiVo to display a command-prompt, as explained in Chapter 7, "Connecting the Universe to Your TiVo," in the section "Getting a Command Prompt on Your TiVo."

The easiest way to install the settings-backup software on your TiVo is to recursively copy that directory from the BATBD CD to a partition on your TiVo disk while you have it in your desktop PC. For now, let's assume that we simply want to put the software in its own directory in the TiVo's **/var** partition. To do so, perform the following steps:

1. Put the TiVo disk in your PC as described in Chapter 3 in the section "Working with TiVo Disks."
2. If your disk is from a Series 1 machine, boot from the BATBD disk using the **s1swap** option so that you can mount and access your TiVo partitions. If your disk is from a Series 2 machine, boot from the BATBD disk using the default boot option by pressing the Enter key at the boot screen.
3. Mount the partition of your TiVo disk that corresponds to its **/var** partition. This will be the partition **/dev/hdX9**, where **X** is the letter corresponding to where you've attached your TiVo drive to your PC. It could therefore be any of the following **/dev/hdb9**, **/dev/hdc9**, or **/dev/hdd9**—or even **/dev/hda9** if you are working with a disk from a Series 2 system. To mount this partition, use a command like the following:

   ```
   mount /dev/hdX9 /mnt/var
   ```

4. Mount the BATBD CD using a command like the following:

   ```
   mount /dev/hdX /mnt/cdrom
   ```

   As in previous examples, replace **X** with the letter corresponding to your CDROM drive. This will be **b** if your CD ROM drive is attached as the slave on your primary IDE interface; **c** if your CD ROM drive is attached as the master on your secondary IDE interface; or **d** if your CD ROM drive is attached as the slave on your secondary IDE interface. If you specified the correct device, you will see a warning message stating that the CD ROM drive is mounted as read-only, which is true and can be safely ignored.
5. Recursively copy the settings-backup directory from the BATBD CD to the **/mnt/var** partition using a command like the following:

   ```
   cp -r /mnt/cdrom/settings-backup /mnt/var
   ```

6. Unmount the BATBD CD and TiVo **/var** partitions by executing the following two commands;

   ```
   umount /mnt/cdrom
   umount /mnt/var
   ```

   After you have finished this process, put the TiVo disk back in your TiVo and reboot it. If you are using a serial connection to your TiVo, attach the cable at this point and start the terminal emulation program on your PC; if you have put your TiVo on the network, wait until it has booted successfully, and then connect to it by using a utility such as telnet.

When the command prompt displays, change directory to the **/var/settings-backup** directory by using the **cd /var/settings-backup** command. You can then create a backup file of your Season Passes and WishLists by executing the following command:

```
./backup > settings.out
```

This command writes a copy of your Season Passes and WishLists to the file settings output, which you should then copy back to your personal computer and put in a safe place. To restore the settings from this file in the future, transfer it back to the TiVo, copy it to the **/var/settings-backup** directory, make that your working directory by using the **cd** command, and then execute the following command:

```
./restore settings.out
```

NOTE

Restoring Season Pass and WishList settings using the **restore** command provided in the **settings-backup** directory does not overwrite any existing Season Passes or WishLists, even if they are identical. Before restoring Season Passes or WishLists, you should first clean up your existing Season Passes or WishLists, perhaps deleting or at least renaming any that may have the same names or that might otherwise conflict with the settings that you are restoring. You can then go through the list and delete any duplicates that you do not want to preserve for some reason.

General Information about Restoring TiVo Data

The next few sections discuss how to restore different types of TiVo backups, whether as part of a system upgrade or simply as part of the repair process for a formerly ailing TiVo. Before attempting a restore for whatever reason, it's important to understand what TiVo data you can restore, and where.

As discussed earlier, image backups of an entire primary TiVo disk always include the entire contents of that disk. Restoring an image backup from one primary TiVo disk to another, therefore, always produces a bootable disk, complete with all of the files required for daily operation. Similarly, structured backups of primary TiVo disks created using the MFS Tools utilities always contain at least the minimum information required to create a bootable disk. However, all TiVos and bootable TiVo disks are not created equal.

One common mistake made by people, who own multiple TiVos or who are restoring backups to a TiVo for some other reason, is to restore a backup of one type of TiVo to another and expect it to work. Series 1 and Series 2 systems use different processors, and therefore run

completely incompatible kernels, libraries, and executables. This is the equivalent of trying to directly execute a Macintosh program on a Windows machine—it simply won't work, ever. There is absolutely no way to restore a Series 2 image to a Series 1 TiVo machine and expect it to work.

Similarly, there are fundamental differences between many of the Series 1 machines, such as the Sony and Phillips/Hughes models. Backups of these systems are therefore also incompatible, even though they may initially appear to work on both types of systems—the TiVo software upgrade process will certainly become confused in the future if, for example, it knows that you have a Phillips TiVo but encounters Sony data when trying to upgrade the system. You can, however, restore backup images to other models of the same series of TiVo from the same manufacturer, as long as sufficient disk space is available to hold the restored data. For example, you can restore a Series 1 Phillips HDR112 image to a Phillips HDR312, and everything will work fine. Similarly, you can restore a backup image from a Series 2 40 MB TiVo to a Series 2 80 MB TiVo with no problems.

TIP

If the first three digits of your TiVo are the same as the first three digits of the TiVo on which a backup was created, the backups are compatible.

NOTE

When restoring backups made on a system with a smaller disk to a system with a larger disk, you will almost certainly want to use the MFS Tools utilities to reclaim the additional space for storing recordings, as discussed later in this chapter in the section entitled "Restoring an MFS Tools Backup to a Larger Drive".

Restoring Image Backups Using dd

Having created an image backup of one of your disks using the **dd** command as explained earlier in this chapter, you can quickly and easily restore this to another disk by essentially reversing the order of the arguments that you supplied to the **dd** command to create the backup. Be extremely careful when supplying arguments to the **dd** command——specifying the wrong disk drive as the target of a restore operation will destroy any data that it previously contained, which would be bad if you accidentally specified the drive containing the Windows or Linux boot information for your personal computer.

TIP

It doesn't technically matter which of the kernels on the BATBD CD you boot with to restore an image created with **dd**, but it should be the same one that you booted from when you created the backup using the **dd** command. (This guarantees that no byte-order conflicts arise.) Since it is useful to mount partitions from the restored disk to verify that the restore completed successfully, I suggest that you use the byte swapped kernel (i.e., the **s1swap boot** option) when creating or restoring a TiVo Series 1 backup image.

Attach the drive that has the image backup you want to restore to your PC (as explained in Chapter 3 in the section "Working with TiVo Disk Drives") and ensure that you also have access to the partition where the backup is stored (as discussed earlier in this chapter in "Finding Room for Backups"). You then can restore the backup image to the new drive using a command like the following:

```
dd if=<filename> of=/dev/hdX bs=32k
```

The <filename> argument should be the full pathname of the file containing the backup image that you are trying to restore. As always, the **X** in **/dev/hdX** reflects the drive letter associated with where you have mounted the target hard drive in your PC.

This command will take a long time. After it completes, try mounting partitions from the restored disk. You should be able to mount **/dev/hdX4**, **/dev/hdX7**, and **/dev/hdX9**. You can try mounting these in turn, using a command like the following:

```
mount /dev/hdXN /mnt/test
```

Mounting a Linux filesystem verifies that the core data structures of the filesystem are valid. You can then immediately unmount them (before trying to mount another) using the command **umount /mnt/test**.

Once you have restored a backup image and have verified that you can successfully access its partitions, what you do next with the disk depends on why you restored the backup image:

- If you were restoring an existing disk image to an identical disk (or even overwriting the original disk) to protect against hardware failures, you could simply put the new disk in your TiVo and begin using it.
- If you were restoring an existing disk image to a larger disk to increase the amount of storage available on your TiVo, you could use the MFS Tools utilities to enable the extra space for use by the TiVo (as explained in Chapter 6, in the section "Replacing an Existing TiVo Disk with a Larger One".

TIP

If you cannot mount any of the partitions from the restored disk image after the restore finishes, try the following:

- Verify that you booted from a kernel that supports access to the TiVo partition tables. On the BATBD disk, these are the **s1swap** boot option for Series 1 machines and the '**s2** boot option for Series 2 machines (also available at the BATBD CD's boot prompt).
- Verify that you are trying to mount the correct partitions. The **ext2** partitions on a TiVo disk are partitions 4, 7, and 9, which you would specify as **/dev/hdXN**, where **X** is the letter for the drive and **N** is the partition number.
- Verify that you did not accidentally create the backup image using a non-byte-swapped kernel, if you are restoring it using a byte-swapped kernel. You can reverse byte-swapping when restoring data using the **dd** command by specifying the **conv=swab** option on the **dd** command-line, as in the following example:

```
dd if=<filename> of=<disk-or-partition> bs=32k conv=swab
```

Restoring Backups Created Using MFS Tools

As discussed earlier, backups of primary TiVo disks created using the MFS Tools utilities are complete backups of all of the portions of a TiVo disk that are critical for successful operation of your TiVo. They also can be complete backups of your TiVo in general, if performed as instructed earlier in this chapter in "Backing Up an Entire TiVo Disk Using MFS Tools."

The next few sections discuss different ways of restoring backups created using the MFS Tools utilities. These largely differ in terms of the command-line options used by the mfstool restore command, but are organized into separate sections in order to make them easier to find and use.

Restoring an MFS Tools Backup Without Adding New Space

In most cases, you will be restoring MFS Tools backups to one or more drives that are larger than the drive that you backed up, and then automatically devoting any additional space on the drive(s) for use when storing recordings on your TiVo. If that's your goal, skip this section and go to either of the next two sections. However, there are cases where you may want to do

a simple restore, either to the same disk that you backed up, another disk of the same size as the original disk, or to a larger disk on which you do not want to automatically allocate additional space for use with your TiVo. Some cases where these circumstances may be useful are the following:

- You suspect that your drive is failing and you can't afford another one at the moment. In this case, after backing up the original drive, you can use the manufacturer's utilities to low-level format the disk so that it marks any failing sectors as bad, then restores a backup of that drive to the drive to try to continue using it.
- You want to preserve your original drive as an on-the-shelf backup and happen to have another drive of the same capacity. In this case, you can put the old disk on the shelf, restore the backup to the duplicate disk, and use the duplicate disk in your TiVo.
- You want to restore your original disk to a larger disk on which you plan to use the additional space for some other purposes—in other words, without automatically allocating the additional space for use by the TiVo. In this case, you can restore the backup, and then use a command such as **pdisk** (discussed in Chapter 4) to manually create new partitions of types such as **ext2** or MFS. You may want to do this as your TiVo hacking skills evolve and you need additional **ext2** partitions to store new software that you've put on your TiVo.

Restoring an MFS Tools backup to a disk without doing anything else is the simplest case of restoring data to a TiVo disk. To do this, first put the disk that you want to restore to in your PC, and boot from the BATBD disk. Next, execute a command like the following:

```
mfstool restore -i <backup-file> /dev/hdX
```

The **mfstool restore** command's **-i** option specifies the full pathname of the file that contains your backup (<backup-file>). The **mfstool restore** command displays the following output during the restore process:

```
Starting restore
Uncompressed backup size: XXXX megabytes
Restoring NNN of XXXX megabytes (M.MM%)
Cleaning up restore.  Please wait a moment.
Restore done!
```

While the **mfstool restore** command is running, the line reading "Restoring NNN of XXXX megabytes (M.MM%)" is continually updated to show the progress of the restore command. Once the command completes, congratulations! You've just created a new TiVo drive from an existing backup file! You can shut down the PC, return the drive to your TiVo, and boot from it successfully.

NOTE

After restoring a backup to a TiVo disk and booting from that TiVo, the TiVo will still contain directory entries for recordings that were present on the TiVo disk when it was backed up, even though the recordings themselves were not preserved in the backup. You should go to the Now Playing screen (TiVo Central -> Now Playing) and delete the entries for these shows, to minimize confusion as to what recordings are actually residing on your TiVo.

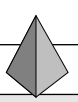
Restoring an MFS Tools Backup to a Larger Drive

By restoring an existing backup to a drive that is larger than the one from which it was made, you can easily add new storage capacity to your TiVo. The **mfstool restore** command provides the **-x** (expand) option so you can restore a backup to a larger drive, and then automatically allocate any additional space on the drive for storage.

To do this, first put the disk that you want to restore to in your PC and boot from the BATBD disk. Next, execute a command like the following:

```
mfstool restore -i <backup-file> -x /dev/hdX
```

The **mfstool restore** command's **-i** option specifies the full pathname of the file that contains your backup (<backup-file>). The **-x** option tells the **mfstool restore** command to expand the set of volumes on the existing disk to allocate any addition storage available on the drive to MFS partitions that the TiVo can use to store recordings.



TIP

If you are expanding the storage capacity of your TiVo beyond 100 MB or so, you should consider increasing the amount of swap space on the drive during the restore process. As explained later in this chapter, this will improve the performance and responsiveness of your TiVo. For more information about increasing swap space, see "Increasing Swap Space on a TiVo Disk," later in this chapter.

The **mfstool restore** command displays the following output during the restore process:

```
Starting restore
Uncompressed backup size: 1309 megabytes
Restoring 14 of 1309 megabytes (1.77%)
```

```
Cleaning up restore.  Please wait a moment.
Restore done!
Adding pair /dev/hdX12-/dev/hdX13
New estimated standalone size: 79 hours (64 more)
```

While the **mfstool restore** command is running, the line reading "Restoring NNN of 1309 megabytes (M.MM%)" is continually updated to show the progress of the **restore** command. As the example output shows, once the restore itself completes, the **mfstool restore** command creates a new pair of MFS partitions (one application region and one media region) to use the additional space that is available on the drive. It then estimates the amount of storage available on the new disk for recordings made at the Basic quality level. The numbers in the example will differ from what you see on your screen because they depend on the size of the drive on which the original backup was made and the size of the drive to which you are restoring the backup.

After the **mfstool restore** command completes, congratulations! You've just created additional storage for your TiVo recordings and created a new drive for use in your TiVo. You can shut down the PC, put the new drive in your TiVo, and boot from it successfully. When your TiVo has booted, you can verify the amount of storage available on the expanded TiVo from the System Information screen (TiVo Central -> Messages & Setup -> System Information).

Restoring an MFS Tools Backup to a Two-Drive TiVo

The **mfstool restore** command also enables you to restore a single backup to a pair of drives. This actually restores the TiVo system information to the first drive, allocates any free space on the first drive for use by your TiVo, and then allocates the space on the second drive for use by your TiVo. This is accomplished by using the same **-x** (expand) option discussed in the previous section—you simply have to specify both of the drives that you want to restore to on the **mfstool restore** command-line.

To do this, first put the disk that you want to restore to in your PC and boot from the BATBD disk. Next, execute a command like the following:

```
mfstool restore -i <backup-file> -x /dev/hdX /dev/hdY
```

The **mfstool restore** command's **-i** option specifies the full pathname of the file that contains your backup (<backup-file>). The **-x** option tells the **mfstool restore** command to expand the set of volumes on the target disks to allocate any additional storage available to MFS partitions that the TiVo can use to store recordings. The drives specified on the command-line must be specified in the order that you will put them in your TiVo: **/dev/hdX** would be the TiVo boot drive, and **/dev/hdY** would be the TiVo slave drive.

TIP

If you are expanding the storage capacity of your TiVo beyond 100 MB or so, you should always increase the amount of swap space on the drive to at least 128 MB during the restore process. You must increase the amount of swap space available, if your new TiVo will contain more than 120 MB of total disk space. As explained later in this chapter, this will improve the performance and responsiveness of your TiVo. You can increase the amount of swap space allocated during a restore by specifying the **-s** <size> option on the **mfstool restore** command line, where <size> is the new amount of swap space that you want to allocate. The default amount of swap space allocated is 64 (MB). A better value for larger drives is 128 (MB). You may even want to increase this to 256 MB, if you will be running additional processes, such as an FTP server, telnet daemon, and TiVoWeb on your expanded TiVo. The **-s** <size> option is discussed in more detail in the "Advanced Restore Options" section later in this chapter.

The **mfstool restore** command displays the following output during the restore process:

```
Starting restore
Uncompressed backup size: 1309 megabytes
Restoring NNN of 1309 megabytes (M.MM%)
Cleaning up restore.  Please wait a moment.
Restore done!
Adding pair /dev/hdb12-/dev/hdb13
New estimated standalone size: 79 hours (64 more)
Adding pair /dev/hdd2-/dev/hdd3
New estimated standalone size: 103 hours (24 more)
```

While the **mfstool restore** command is running, the line reading "Restoring NNN of 816 megabytes (M.MM%)" is continually updated to show the progress of the **restore** command. As the example output shows, once the restore itself completes, the **mfstool restore** command creates a new pair of MFS partitions (one application region and one media region) to use the additional space that is available on the first drive, it creates one or more pairs of MFS partitions on the second drive, and then estimates the amount of storage available on the new disks for recordings made at the Basic quality level. The numbers in the example will differ from what you see on your screen because they depend on the size of the drive on which the original backup was made and the size of the drive to which you are restoring the backup.

After the **mfstool restore** command completes, congratulations! You've just substantially increased the amount of space available on your TiVo for storing recordings. You can now shut down the PC, put the new drives in your TiVo, and boot from them successfully. When your

TiVo has booted, you can verify the amount of storage available on the expanded TiVo from the System Information screen (TiVo Central -> Messages & Setup -> System Information).

Advanced MFS Tool Restore Options

The **mfstool restore** command provides a variety of options that you can use to customize different aspects of your disk layout when restoring a backup created using the MFS Tools utilities. Each of these advanced capabilities is activated by specifying an additional option and any associated values on the **mfstool restore** command line. The following sections discuss these advanced options and the situations in which you might want to use them.

Increasing Swap Space on a TiVo Disk

Like most modern computer systems, your TiVo uses swap space to increase the apparent amount of memory available to the system. Swap space is a specially formatted portion of the disk that the version of Linux running on your TiVo can use as temporary space for processes that are active on your TiVo, but which are waiting for some event to occur in order to continue. Moving paused or lower-priority processes out of main memory and temporarily storing them in the swap space enables the processes that are actively running to use the memory that the paused or lower-priority processes would otherwise have used.

Swap space is also used during certain parts of the boot process. The most important of these is the step where the TiVo verifies that the partitions on your hard drive are usable. Known as a *disk consistency check*, this requires that the process that is doing the check build a table in memory that contains detailed information about your disk drives. The size of this table varies depending upon the size of your disk drives. If your TiVo contains more than 120 MB of total disk space, this process will need more than the default 64 MB of swap space provided on TiVo disk drives.

To increase the amount of swap space that is allocated on your hard drive during a restore, you can use the **-s** <size> command-line option to the **mfstool restore** command. The <size> value is an integer value up to 511, and indicates the number of megabytes of swap space that will be created during the restore process. If you ever plan to have 120 MB or more disk space in your TiVo, you should allocate a minimum of 128 MB of swap space. A better value is 256 MB, especially if you may run processes such as a telnet daemon or the TiVoWeb server on your TiVo. You cannot easily change the amount of swap space that is allocated on an existing disk, so selecting an appropriate value during the restore process is a good idea.

Increasing the Size of the /var Partition on a TiVo Disk

Like all Linux and Unix systems, the TiVo uses a hierarchical set of files and directories to store system information. On Linux and Unix systems, disk partitions used by the system for storing files are mounted on directories in the filesystem, which are known as mountpoints, which are just standard Linux directories that can be recycled as entry points to additional storage. After a partition has mounted on a directory, the storage in that partition then is used by any files and directories created in that directory or any of its subdirectories. By default,

TiVos only have two Linux partitions: the root partition, which is mounted at the directory named / (the "root" of the Linux filesystem), and the **/var** partition, which is used to store applications and variable data such as system logs, temporary space for creating non-critical files, upgrades and advertisements that you have received from TiVo headquarters, and so on.

The default size of the **/var** partition on a TiVo is 128 MB, which is suitable for "normal" use. However, if you decide to install additional applications on your TiVo, you may find that 128 MB is somewhat constraining. The **mfstool restore** command provides the **-v** <size> option to specify the size of the **/var** partition created when restoring a TiVo from backups. The <size> value that you specify should be an integer value that is interpreted as the number of megabytes that you want to have allocated to the **/var** partition. If you are planning to do some serious hacking on your TiVo or simply want to prepare for doing so in the future without reformatting your disk, you should specify a higher value when using the **mfstool restore** command. A value of 256 MB should be sufficient for most purposes, though you can certainly use higher values if desired. You can always use the **pdisk** command (discussed in Chapter 4) to manually create additional partitions in the future and mount them wherever you want, as long as your disk contains available space.

WARNING

Most TiVo systems running TiVo software releases newer than Version 3.0 use hashing to verify the contents of many of the default files used during system startup. You may not be able to mount additional partitions unless you use some of the techniques explained in Chapter 7, in "Getting a Command Prompt on Your TiVo," to modify the boot sequence and environment on your TiVo. This is probably OK, since you will only be adding and mounting new partitions if you're doing some serious TiVo hacking, but "forewarned is four armed," as they say ;-).

Optimizing the Partition Layout on a TiVo Disk

One of the governing factors in the performance of any computer system is the amount of time that it takes your disk drive to read and write information to and from your disk. Regardless of how fast your computer system is electronically, it is still gated by these physical operations, which require moving the heads of the disk to the location where the data is (or is to be) stored. In computer terms, the head movement necessary to locate data is known as *seeking*; the amount of time required to locate a unit of data on the disk is generally referred to as the *seek time*. Minimizing seeking and seek time when using a computer system is an important, but easily overlooked aspect of system performance and optimization.

Newer TiVos, such as the DirecTiVo and newer Series 2 models, use a slightly different partition layout than original Series 1 systems, in an attempt to minimize the movement of your disk's read/write heads and reduce the average seek time. MFS partitions are located on the outside of the disk to minimize the amount of head movement necessary to access them, while

the Linux partitions containing application data are put in the center of the disk, since applications are started/stopped less frequently than video data is read and written.

The **mfstool restore** command's **-p** option restores data to the new disk as described in the previous paragraph. Although today's disks are substantially faster than the drives in, for example, original Series 1 machines, using this option can provide a general performance improvement, and may also reduce the amount of noise generated by your disk as a consequence of having to seek less.

Changing the Block Size When Restoring a TiVo Disk

Different types of computer filesystems use different block sizes when allocating disk space and storing data. The block size of a filesystem is the smallest unit of space that the filesystem can allocate. If a filesystem's block size is 4 MB and you need to store 1 byte of data, the filesystem must still allocate one 4 MB block to store that data. At first, it seems silly to use larger block sizes. Why not simply use a tiny block size and allocate as many blocks as necessary to store a given amount of data?

The reason for larger block sizes is attributed to the system overhead related to allocating and managing blocks in the filesystem. Using a smaller block size means that the system must allocate a greater number of blocks to store files, as well as manage a much higher number of blocks within the filesystem, tracking whether each is used or free (i.e., can be allocated). The data structures within a filesystem that track which blocks are associated with that file, and in what order, must also be correspondingly larger. Selecting an appropriate block size is especially important in systems that routinely create and delete large files, as your TiVo does when recording and deleting videos. Larger block sizes tend to "waste" more disk space because the final block of any file is rarely completely filled with data, but significantly less effort is involved in allocating and using the larger blocks of data required for classically larger video files.

As a more detailed example, assume that a filesytem's block size was 16 MB and the TiVo was trying to store a recording that required 20 MB of disk space. This would require two blocks and would completely fill the first one but waste 12 MB of disk space in the second. In order to keep track of the recording, the TiVo would keep only information about the two blocks in memory. On the other hand, with a block size of 4 MB, the recording would require 5 blocks but no disk space would be wasted. However, the TiVo would have to keep information about 5 blocks in memory to keep track of that recording.

The **mfstool restore** command's **-r** <value> option enables you to specify the block size used within the MFS partitions, created when restoring a backup file created using the **mfstool backup** command. Acceptable values for this option are 0 (corresponding to a 1 MB allocation unit), 1 (corresponding to a 2 MB allocation unit), 2 (corresponding to a 4 MB allocation unit), 3 (corresponding to an 8 MB allocation unit), and 4 (corresponding to a 16 MB allocation unit). In general, the larger the allocation unit value, the less RAM will be used (making the TiVo more responsive when displaying menus, changing and processing options, and so on), but the greater the amount of storage that may be wasted when storing recordings.

The default block size used in partitions created by your TiVo is 1 MB. This was a good value to use with the relatively small drives and small amount of memory used on early TiVos.

However, today's larger, faster drives make using a larger block size an attractive option for improving performance.

Changing Byte Order When Restoring a TiVo Disk

As mentioned over and over throughout this book, the Series 1 models use a big-endian PowerPC processor. The byte order in Series 1 filesystems is therefore different from the byte order used in Series 2 systems (and in desktop Linux systems). The initial version of the MFS Tools utilities didn't concern itself with byte-order because there was only one at the time. However, with the introduction of Series 2 systems, Version 2 of the MFS Tools utilities introduced two options for manipulating the byte-order in a backup file created using the **mfstool backup** command.

You may wonder why this is important. After all, you can still use Version 1 of the MFS Tools utilities to restore backups that it created, and everything will "just work." Unfortunately, this is only true if the kernel that you've booted from uses the same byte order as the one that you were running when you created the backup. A further wrinkle is that Version 2 of the MFS Tools utilities added many other features that increased the power and scope of these utilities. As with so many things, "newer is better."

Tiger, the TiVo hacker who wrote the MFS Tools utilities, took this into account when creating Version 2 of the MFS Tools utilities, adding options that let you permute your backups any way you'd like during the restore process. Ordinarily, the **mfstools restore** command attempts to automatically determine whether byte-swapping should be accomplished by examining the byte order on disks such as **hdb**, **hdc**, and **hdd**. The **-b** option disables this auto-detection and forces a restore to be performed without byte-swapping. This option is mandatory when restoring a backup of a Series 2 machine that was created with Version 1 of the **mfstool backup** utility. The parallel **-B** option disables auto-detection and forces a restore to be accomplished with byte-swapping. This option is mandatory when restoring a backup of a Series 1 machine, if you are running a non-byte-swapped kernel, such as the BATBD kernel for Series 2 systems.

TIP

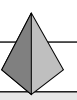
Byte-swapping differences can be a pain in the posterior. When restoring a TiVo backup using **mfstool restore**, you should always boot with a kernel that provides access to the TiVo partition table so that you can query a restored disk's partition table to determine if it was restored correctly. If you are restoring a Series 2 backup, the BATBD disk provides a single boot option because the Series 2 systems don't use byte-swapping. If you are restoring a Series 1 backup, boot with the BATBD's **s1swap** option. If you've gotten the byte-order wrong during your first restore, you may need to restore the backup again with the correct byte-swapping options enabled. However, you can be sure that the restored disk will be readable—and hopefully bootable——on your TiVo.

Verifying TiVo Disks Restored Using MFS Tools

After restoring a backup to a new disk that you want to use in your TiVo, you should always verify that the backup looks reasonably correct by examining the partition table of the new disk using the **pdisk** command, and also by mounting one or two partitions from the restored disk. Moving drives from your PC to your TiVo and back again is something of a pain, and the TiVo is light on feedback if you try to boot from a non-bootable drive. If there's something wrong with your restore, in the best case your TiVo boots to the gray "Just a few more seconds..." screen, and sits there until the end of the universe. In the worst case, you hear lots of disk noise and the red light displays on the front of the TiVo. What the heck does all that mean? In a nutshell (Sorry, O'Reilly guys!), it means that your restore was unsuccessful for one reason or another.

You can verify that the disk you restored your data to has a valid partition table by executing the command **pdisk -p /dev/hdX**, where **X** is the drive letter corresponding to the restored disk. If this command displays a partition table, the restored disk is at least valid and there's a good chance that your TiVo will boot from it. If you don't see a valid partition table and instead see a message that the drive does not have one, make sure that you've booted with a kernel that provides access to the TiVo partition table. If you have restored a Series 2 backup, boot from the BATBD disk's **s2** kernel by pressing Return at the BATBD boot screen. If you have restored a Series 1 backup, boot from the BATBD disk's **s1swap** kernel by entering this value at the BATBD boot screen. If you still cannot see the partition table and are backing up or restoring a Series 1 TiVo disk, try experimenting with the byte-swapping options discussed in this chapter in "Changing Byte Order When Restoring a TiVo Disk."

Once you can see the partition table, you should be able to mount one of **/dev/hdX4** or **/dev/hdX7**, and you also should be able to mount **/dev/hdX9**. As mentioned earlier, the **mfstool backup** command only backs up the currently active root partition from your original TiVo disk to reduce the size of your backup file when doing backups. For this reason, when trying to mount **/dev/hdX4** and **/dev/hdX7**, one of these will fail because it was the inactive root partition.



TIP

You can use the **bootpage** command to identify the default root partition on a drive. For example, the command `bootpage /dev/hdb` displays the default root partition on the drive **/dev/hdb**.

You can try mounting each of these in turn by using a command like the following:

```
mount /dev/hdXN /mnt/test
```

Mounting a Linux filesystem verifies that the core data structures of the filesystem are valid. You can then immediately unmount them (before trying to mount another) by using the command **umount /mnt/test**.

CLONING THE ROOT PARTITION ON A RESTORED TIVO

I always prefer to have two root partitions available on any restored TiVo disk. In theory, you will only need the second root partition after a software upgrade has populated it with a new kernel fresh from TiVo HQ. However, in reality, a disk problem could corrupt your existing root partition, leaving you with an unbootable TiVo disk. The truly wizardly among us would just clone the root partition from another TiVo disk using **dd**, but why use your powers if you don't have to?

After restoring a TiVo disk, try mounting both **/dev/hdX4** and **/dev/hdX7** to determine which of them actually contains a valid root filesystem (or use the **bootpage** command to query the disk to identify the active root filesystem). After you know which one is valid, you can clone it to the other using the **dd** command. For example, if **/dev/hdd4** is the valid root partition on a restored TiVo disk and **/dev/hdd7** is just an unmountable hunk of ones and zeroes, you can clone **/dev/hdd4** to **/dev/hdd7** by using the following command:

```
dd if=/dev/hdd4 of=/dev/hdd7 bs=32k
```

You then should be able to mount **/dev/hdd7**. Try it. You'll like it! It's like a free insurance policy. The TiVo won't always be able to switch boot partitions by itself (you may have to do so using the **bootpage** command), but at least the data will be there.

Connecting Backup and Restore Commands Using a Pipe

If the disks on your personal computer do not have the space to create a complete or partial backup of an existing TiVo disk, you can clone an existing disk to another disk by using the **mfstool backup** and **mfstool restore** commands together. You can even use this process to add another disk at the same time. As touched upon earlier, you can do this by taking advantage of the ability of Linux to connect the output of one program to the input of another. In Linux terms, this is done using a pipe, which is represented on the command-line as the "|" symbol. In this way, you can use the **mfstool backup** command to back up your existing disk, writing the backup file to standard output, which the corresponding **mfstool restore** command will then use as its standard input.

All of the command-line options for the **mfstool** backup and **mfstool restore** commands work in exactly the same way when writing to a pipe, with the exception of the options that specify input and output files. When writing to a pipe on a Linux or Unix system, the input and output files are specified as a single dash (-). An example of a command that would back

up the contents of the drive attached to your PC as **/dev/hdb** and simultaneously restore that backup to a larger drive attached as **/dev/hdc** would be the following:

```
mfstool backup -o - /dev/hdb | mfstool restore -i - -x /dev/hdc
```

You can use this same approach to do more sophisticated backups and restores, such as when you want to clone your existing drive to a larger one but preserve all of your existing recordings, as in the following example:

```
mfstool backup -o - -a /dev/hdb | mfstool restore -i - -x /dev/hdc
```

As a final example, here is a sample set of backup and restore commands that do a compressed backup of two TiVo drives, and then restore them to a set of two (presumably larger) TiVo drives, thereby increasing the amount of swap space to 256 MB and increasing the size of the **/var** partition to 512 MB. This command is split across two lines for readability in this book (you would ordinarily enter it on a single line, removing the backslash that separates the two commands):

```
mfstool backup -o - -a -s /dev/hda /dev/hdb | \
   mfstool restore -i - -x -s 256 -v 512 /dev/hdc /dev/hdd
```

NOTE

This command would not work correctly if you were using the BATBD disk's byte-swapped kernel because the **/dev/hda** drive is not byte-swapped, while the others are. In this case, a better approach would be to attach the two original TiVo drives so that they show up on your Linux systems as **/dev/hdb** and **/dev/hdc**, and then attach one of the new drives so that it shows up as **/dev/hdd**. If the new drive is large enough, you could back up both drives to the new larger drive, shut down the system, remove the old drives, and attach the new drive as **/dev/hdb** or **/dev/hdc**. Then use the procedures discussed in Chapter 6 to add the second drive to the new drive that you restored to in the first step. Clear as mud?

Changing TiVo Operating System Versions Using Backups

The version of the TiVo operating system contained on a bootable TiVo disk is inherently backed up as part of any unstructured backup of an entire TiVo disk, or as part of any structured backup created using the MFS Tools utilities. Therefore, restoring that backup image also restores that version of the operating system to the disk on which you are restoring the data.

As mentioned at the beginning of this chapter, the TiVo's ability to be remotely upgraded is both an advantage and a potential problem. If you are simply interested in running the latest

and greatest version of the TiVo software and aren't necessarily interested in getting a command prompt on your TiVo, logging in over the network, or running TiVoWeb, automatic upgrading of the software is a great feature. One morning you wake up, and your TiVo has an enhanced user interface, improved audio or video recording capabilities, or perhaps additional commands and options. It's like Christmas!

On the other hand, if you have carefully hacked your TiVo to provide network access, automatically start the TiVoWeb application for remote access to your TiVo, or if you simply want to take advantage of things like backdoors and related special commands and modes, you would be dismayed to find that your TiVo has been enhanced behind your back, and typically in a

SOFTWARE PROTECTION AGAINST UPGRADES

There is no fool-proof way to protect against software upgrades with the exception of disconnecting your TiVo from the phone line or network connection through which it obtains both programming and updated guide data. However, certain versions of the TiVo operating system support boot parameters that prevent software upgrades in most cases. Passed to the TiVo as boot parameters that are stored in the TiVo's bootpage, these options will protect against all but the most aggressive upgrades.

Your TiVo's boot parameters are set using the **bootpage** command discussed in Chapter 4. Before updating the boot parameters, you'll want to check what they currently are, using a command like the following:

```
bootpage -p /dev/hdX
```

Replace the **X** with the letter corresponding to where the TiVo disk is located in your PC. This will display something like the following:

```
root=/dev/hda4
```

Once you know your current boot parameters, protecting your TiVo against most software updates is done using the following command:

```
bootpage -P "root=/dev/hdYY upgradesoftware=false" -C /dev/hdX
```

Replace the YY with the appropriate letter and number given in the output of the previous **bootpage** command, and replace the **X** with the letter corresponding to where the TiVo disk is located in your PC. After completing this command, only major version TiVo software updates should get through, if anything.

The potential downside of this approach is that the TiVo will continually notice that your TiVo is not upgraded, and may continue to download the upgrades even if you're not installing them. The folks at TiVo may get cranky about this and could disable your account until you talk to them. The best protection against upgrades is to back up your hacks so that you can easily restore them if your TiVo is forcibly upgraded.

way that stops your careful hacks from working. (In some cases, they will even be deleted!). For this reason, it is important to always have a backup available that contains a snapshot of your TiVo when it was customized just the way you wanted it. If your TiVo has been upgraded and you find that you prefer the hacked version with an older TiVo software release to the new, unhacked TiVo software release, you can always restore your old system from backups, returning your system to the state that you desire.

Dumping TiVo Data to Videotape

For computer systems, backups typically refer to external media that contains copies of critical system programs, system configuration data, and user data. In this last sense, your TiVo provides a built-in mechanism for backing up its equivalent of user data, which consists of the programs that you've recorded. Earlier sections of this chapter explained how to back up these recordings as a part of an entire system backup, which is fine if you also want to restore your entire system. However, since most of us still have an old dusty VCR sitting around somewhere from our pre-TiVo days or for the occasional video rental, it's also useful to back up some of your more precious recordings onto videotape, or for sharing them with friends who may have missed a specific episode of your favorite show. Though videotapes are somewhat retro in the age of the DVR, they're still a useful exchange medium—something like what floppy disks are to today's personal computers.

Your TiVo provides built-in support for backing up recordings onto videotape through the Save to VCR command, which is available on the Now Playing screen for any recording. After connecting your VCR to your TiVo (explained in great detail in the TiVo Installation Guide that came with your TiVo), make sure that your VCR is set to record from the input source to which you attached your TiVo. Now put a videotape in the VCR.

NOTE

If you don't have your TiVo Installation Guide handy and never bothered to connect your VCR after getting your TiVo, you can download PDF versions of the "TiVo Installation and Viewer's Guide" for your TiVo from http://www.tivo.com/4.2.2.asp. (A link to this page is found on the Customer Support portion of the TiVo Web site if the TiVo Web site changes.) Viewing PDF files requires that you have installed Adobe Acrobat Reader on your system, which you can freely download from http://www.adobe.com/products/acrobat/readermain.html.

After selecting the recording that you want to save to videotape from the Now Playing screen, highlight the Save To VCR screen option, and press Select. This displays an auxiliary screen from which you should then select "Start saving from the beginning." This means your TiVo will begin to play the video from the beginning—regardless of whether you've been watching

it and are therefore positioned in the middle of the recording. At this point, your TiVo gives you about 10 seconds to press the Record button on your VCR before actually starting to play the recording on your TiVo.

TIP

Although the TiVo displays the recording in real-time so that any VCR can record it successfully, any of the TiVo **backdoor** commands that you already may have activated are available while the recording is playing. If you have activate the 30-second-skip backdoor (described in the section of Chapter 2 entitled "Using Select-Play-Select Codes"), you can use this command (or even the standard fast-forward commands) to skip or fast-forward through any sections of the recording that you may not want to preserve on videotape.

Forbidden Topics Like Video Extraction

As mentioned in the introduction to this chapter, video extraction is one of the touchier subjects in TiVo-land. This is not surprising, because the entertainment industry has one of the highest profit margins in the known universe. One has only to look at the industry's reaction to MP3s and other forms of online audio to see a comic-tragedy of greed unfold. (I'm lumping audio and video entertainment together because they are essentially the same thing, digitally.) If the recording industry had invested the resources spent in trying to stomp out online music, in retailing it fairly and easily, they could have been making money rather than enemies.

You can find a good deal of software and related Web sites on the Internet that describes how to back up personal recordings from your TiVo to other computer systems. Software packages such as ExtractStream (http://www.9thtee.com/extractstream.html) and TyStudio (http://dvd-create.sourceforge.net/tystudio/index.shtml) are rumored to be some of the best, although they are currently oriented towards the 1 TiVos models. The heightened security that is built into the Series 2 makes it more challenging to back up videos from these systems. Although you could apply the same techniques to your Series 2 system after hacking it to get a command prompt as described in Chapter 7. Using the Linux **netcat** utility to send a stream of video data to another computer system is fairly straightforward, and yet another example of the power of Linux. Mapping TiVo MFS filesystem identifiers (fsids) to the specific recordings that you want to back up is easily done through a variety of scripts that are also available on the Internet. If you have sufficient free space on your TiVo, you can even dump specific fsids to files using the **mfs_stream** utility and then create separate audio and video streams using a utility such as **vsplit**. However, I am not going to discuss any of these.

One sure-fire mechanism for backing up videos from your TiVo is to purchase a TV tuner card that will work with your personal computer system. These work with Windows, Linux, and Macintosh systems, and typically come with software tailored to the card that enables you to

display incoming video signals in a window on your PC screen. You can put the card in your PC, use the PC as a display device for the TiVo by connecting the TiVo's Cable Out or S-Video connectors to it, and then use the software provided with the card to capture the video and audio that is being displayed on your PC. TV Tuner cards are available online or on eBay at prices ranging from $20 to hundreds of dollars, depending upon quality, sophistication, and the software packages that are included with them.

Not surprisingly, the Open Source movement has produced an excellent video display and recording package in video display software such as xawtv. Many TV tuner cards use BrookTree chips, which are well supported in the BTTV drivers, which are included with most Linux distributions. They are also available from http://bytesex.org/bttv.html. The xawtv software for displaying video input from a tuner card under Linux (my personal favorite) is available from http://bytesex.org/xawtv/index.html. xawtv also is easy to set up and use, and works with a large number of available TV Tuner and related video input cards. A list of supported cards is included in the source code, though many more have probably been used successfully. If you are using xawtv with a BrookTree-based TV tuner card, you will probably want to get the specific set of fonts that it uses, which are available at http://bytesex.org/xawtv/tv-fonts-1.1.tar.bz2.