ZIFF-DAVIS THE COBB GROUP

October 1997 • $11.50
Vol. 3 No. 10

INSIDE SOLARIS™

Tips & techniques for users of SunSoft Solaris

IN THIS ISSUE

Booting from an alternate hard drive

By Boris Loza

Imagine that one morning your system doesn't come up after a regular planned shutdown. A root file system is corrupt, or some of the files (*/etc/passwd*, */usr/bin/login*, etc.) are damaged. Your company's operations are interrupted, and your office is crowded with people expecting you to resolve the problem immediately!

You can easily prepare yourself for such dilemmas by creating an alternate boot device. You simply need a spare hard drive with enough capacity to hold your root file system. It's not necessary that the hard drive have the same geometry (i.e., it need not have the same number of heads, cylinders, or sectors per track). In this article, we'll show you step-by-step how to create this alternate hard drive, then we'll show you how to boot off it.

Attach a hard disk...

First of all, you must add a disk drive to your system. Once your disk is physically attached to the computer, you'll perform a `boot -r` (or you can `touch /reconfigure` before shutting down) so that Solaris will detect your new disk drive and create the appropriate nodes in the */dev/dsk* and */dev/rdsk* directories.

Next, use the `format` utility to create any partitions you'll want on the disk drive. Then, you must create the appropriate file systems on your new hard disk drive, using `newfs` or `mkfs` (though using `newfs` is simpler). Finally, mount your alternate root partition to a temporary mount directory (like */mnt*). Now you're ready to make a backup root partition.

...or use another slice

Another option is to make an auxiliary root partition on an unused slice of one of the disk drives you already have. However, most of us don't have many spare slices lying around. If you do, you simply need a slice large enough to hold the entire root partition, though you'll probably want to include some additional space, just in case.

If you elect to use another slice on an existing hard drive, you'll probably want to select a hard drive other than the one holding your current root partition. After all, one reason you may want to boot from an alternate root is because the drive holding your primary root partition has failed and you need to recover from it.

Create a boot block on the alternate root slice

Let's assume, for example, that the device name for your alternate boot device is */dev/dsk/c3t3d0s0*. We

SOFTBANK ZIFF-DAVIS

INSIDE SOLARIS
Tips & techniques for users of SunSoft Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by The Cobb Group.

Prices

U.S.	$115/yr ($11.50 each)
Outside U.S.	$135/yr ($16.95 each)

Phone and Fax

US toll free	(800) 223-8720
UK toll free	(0800) 961897
Local	(502) 493-3300
Customer Relations fax	(502) 491-8050
Editorial Department fax	(502) 491-4200
Editor-in-Chief	(502) 493-3204

Address
Send your tips, special requests, and other correspondence to:

The Editor, *Inside Solaris*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220
Internet: inside_solaris@zd.com.

For subscriptions, fulfillment questions, and requests for group subscriptions, address your letters to:

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220
Internet: cobb_customer_relations@zd.com

Staff

Editor-in-Chief	Marco C. Mason
Contributing Editors	Al Alexander
	Jerry L.M. Phillips
	W. Dean Stanton
	Boris Loza
Production Artists	Marguerite Winburn
	Natalie Strange
Editors	Karen S. Shields
	Joan McKim
	Mark Ray
Publications Coordinator	Linda Recktenwald
Product Group Manager	Michael Stephens
Circulation Manager	Mike Schroeder
Managing Author	Eddie Tolle
Publisher	Mark Crane
President	John A. Jenkins

Back Issues
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $11.50 each, $16.95 outside the US. We accept MasterCard, Visa, or American Express, or we can bill you.

Postmaster
Periodicals postage paid in Louisville, KY and additional mailing offices.
Postmaster: Send address changes to:

Inside Solaris
P.O. Box 35160
Louisville, KY 40232

first want to mount this file system at */mnt*, so we'll enter the command

```
$ mount /dev/dsk/c3t3d0s0 /mnt
```

Now comes the most important part: We're going to create a bootable slice, so we must install a UFS boot block on this slice. The UFS boot program is platform-dependent and resides in a platform-dependent directory. Since you can use the `uname -i` command to find the platform name, you can move to that directory using the command

```
$ cd /usr/platform/'uname
➥-I'/lib/fs/ufs
```

Once you're in the right directory, you can use the `installboot` program to create the boot block on your drive. Please note that the syntax is slightly different for the SPARC and x86 versions of Solaris. For a SPARCstation, the syntax is

```
$ installboot bootblock
➥ raw-disk-device
```

where ***bootblock*** is the name of the code that boots Solaris from a disk slice and ***raw-disk-device*** is the name of the slice you want to boot from. Since the name of the boot block file is `bootblk`, and we're installing the boot block to */dev/rdsk/c3t3d0s0*, our command for a SPARCstation would be

```
$ installboot bootblk
➥ /dev/rdsk/c3t3d0s0
```

If you've read the article "Subdividing Your Hard Disk for Solaris x86" in the June 1996 issue, you'll remember that Solaris adds an extra layer on an x86-based PC because the standard BIOS for the PC supports only four partitions on a disk drive, while Solaris supports more. To handle this additional layer, the `installboot` program must install two boot blocks: one to tell the x86 BIOS how to read Solaris' extended partition table, and the standard boot block in the slice that we're making our alternate root drive. This setup is reflected in the syntax of the `installboot` command for a Solaris x86 computer

```
$ installboot pboot bootblock
➥ σraw-disk-device
```

where ***pboot*** is the name of the code that boots the appropriate x86 partition and the ***bootblock*** and ***raw-disk-device*** parameters have the same meaning as they do for SPARCstations. The partition boot code is named `pboot`, so you can install the x86 boot block with the command line

```
$ installboot pboot bootblock
➥/dev/rdsk/c3t3d0s0
```

Transfer your current root directory to your alternate root

After you've successfully installed the boot block to the alternate root slice, you need to copy the root file system. In this example, we're using the `ufsdump` and `ufsrestore` commands, but you could just as easily use `tar` or `cpio` to do the trick. To copy the root file system, just enter the following command:

```
$ ufsdump 0f - / | (cd /mnt;
➥ufsrestore xf -)
```

The - symbol tells `ufsdump` and `ufsrestore` to use the standard input and output devices respectively. This allows us to use `ufsdump` and `ufsrestore` in a pipeline to perform the transfer in one easy step, rather than having to back up the file system to a tape, then restore to the alternate root slice.

Finally, you must modify the */etc/vfstab* file on the alternate root partition to tell it to mount your new root slice as the root partition. Otherwise, the system will get confused as it attempts to boot up, and all your efforts will be in vain. To make this modification, edit */mnt/etc/vfstab*, find the entry for the / file system, and change it to reflect the alternate root slice. For our example, if the default root slice is named */dev/dsk/c0t0d0s0*, we'll change it to */dev/dsk/c3t3d0s0*. The entry shown in blue in Figure A is the definition for the root slice, and it reflects the original root slice. Change the disk slice names, as shown in Figure B, to let Solaris boot from our alternate root system.

Booting from your alternate root slice on a SPARC...

Now we're ready to boot from our new alternate root slice. To do so, dig out the appropriate handbook for your version of the operating system. For a SPARC-based computer, when you shut down and see the > prompt, type *n*. Then the boot PROM will present you with the *ok* prompt, so type *reset*. Your computer will start loading Solaris (if `autoboot` is enabled). After the word *Testing* appears on your screen, abort the `boot` process by pressing the L1-a keys simultaneously. (The L1 key is labeled STOP on type 5 keyboards.) You should again see the *ok* prompt.

Next, you need to find the SCSI addresses, so enter the `probe-scsi` command or (if your boot PROM offers it) the `probe-scsi-all` command. Since the boot PROM deals directly with hardware devices in the system, each device has a unique name representing the type of device and where that device is located in the system-addressing structure. The following example shows a full device path name for the boot PROM:

```
/sbus@1f,0/esp@3,400000/sd@3,0:a
```

In this example, 1f,0 represents an address on the main system bus, 3,400000 is a slot number (slot 3), and 3,0 is a SCSI target and logical unit number, because the drive is attached to a SCSI bus at target 3, logical unit 0. So the full device path name mimics the hardware addressing method used by the system to distinguish between the different devices.

Using the `probe-scsi` command, you can find the device path name for your new disk drive. Then you can specify a device alias that's a shorthand representation of the device path. To display all current device aliases, type `devalias`. Since your primary boot device alias is *disk1*, you may want to use an alias of *disk2* for your alternate root slice. To do this, just type

```
ok devalias disk2
➥/sbus@1f,0/esp@3,400000/sd@3,0:a
```

User-defined aliases are lost after a system reset or power cycle. To create a permanent alias, use the `nvalias` command. Now if you want to boot from a backup device at the *ok* prompt, execute the following command:

```
ok boot disk2
```

For a thorough explanation of the `boot` procedure, please consult the *OpenBoot Reference* or the Solaris 2.x System Administration manuals.

...and on an x86-based computer

For Solaris x86 users, the booting procedure is considerably different, primarily because the system BIOS doesn't have the same functionality as the boot PROM on a SPARCstation. If you installed Solaris and let it take over all your disk drives, then you must boot from an installation floppy so you can get to the boot menu shown in Figure C. If you left another OS on your computer when you installed Solaris, then it already installed the boot menu program

Figure A

```
#dev to mount        dev to fsck           mnt FS    fsck  mnt@  mnt
#                                           at type  pass  boot  opts

/dev/dsk/c0t0d0s0  /dev/rdsk/c0t0d0s1     / ufs     1      no     -
```

If we don't change the device names to reflect the new root location, Solaris will crash while booting.

Figure B

```
/dev/dsk/c3t3d0s0  /dev/rdsk/c3t3d0s1     / ufs     1      no     -
```

Please note that the first column is the block device (i.e., in /dev/dsk), while the second specifies a raw device (in /dev/rdsk).

Figure C

```
  Solaris for x86 - Driver Update 10    Version 1



          Solaris/x86 Multiple Device Boot Menu

      Code  Device Vendor   Model/Desc         Rev
      ================================================
       10   NET    3COM/5x9 I/O=6000 IRQ=15
       11   DISK   SEAGATE  ST43400N           1022
       12   DISK   SEAGATE  ST43400N           1028
       13   CD     TOSHIBA  CD-ROM XM3401TA    0283
       14   TAPE   CONNER   CTMS  3200         7.08
       15   CD     NEC      CD-ROM DRIVE:222   3.0i

       Enter the boot device code: _



  22
```

Use the boot menu to select the drive you want to boot from.

for you. In either case, you simply select the appropriate boot device from the menu.

Alternatively, some SCSI cards and/or BIOSes allow you to select the drive to boot from. You can use the instructions with your SCSI card or motherboard to instruct your computer to boot from your new drive.

Miscellaneous notes

Please keep one thing in mind: You're building your alternate root slice so that you can recover your system in the event something breaks. To this end, be sure that you have access to all the tools you're likely to need when your system crashes.

For example, if the hard drive containing your default root slice fails, you'll need to install a new hard drive, slice it up, make new file systems on the drive, then restore any data from your backups. Therefore, you'll need at least the `format` and `ufsrestore` programs.

If your primary root slice becomes slightly damaged, having an alternate root slice can be a lifesaver—but you'll need a copy of `fsck` to help fix the problems. Be sure it's available, or you'll be in trouble.

You may want to keep a copy of your most important tools on your alternate root slice in addition to their usual location in your file system. After all, those files may become corrupted in the same event that caused the problems with your default root slice.

Conclusion

You never know when your system is going to give up the ghost. If the root partition of your hard drive becomes corrupted, you may have to spend time reinstalling Solaris before you can attempt to restore your precious backups. If you have an alternate root slice, however, things are much simpler: You should be able to boot up on your alternate root slice and perform any administrative tasks before recovering your computer. ❖

Boris Loza, Ph.D., is a systems administrator for Signature Vacations in Toronto, CA. He earned his doctorate in computer science in Russia.

Capturing print jobs before they're printed

By Al Alexander

When trying to troubleshoot Solaris printing problems, you'll find it's often desirable to capture the output from the Solaris print spooler before the spooler sends the job from the spool area to the printer. Once you have the spooler output, you can examine it for clues to help solve your printing problems. You can approach the solution in several ways. In this article, we'll discuss one of the easiest methods.

Capturing output

This approach to capturing print spooler output relies on disabling a printer with the `disable` command, then locating the file that the print spooler would print to if it were allowed to continue. That file may be in one of several locations, depending on the method in which you submit the print job and the file type. Even so, it's generally easy to locate the spool file.

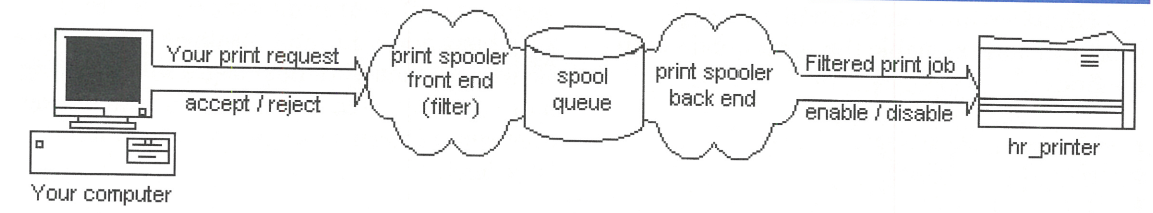
The first step in this process is to disable the printer with the `disable` command. Disabling the printer lets the print spooler accept the print job and perform any filtering required on the print job. However, since the printer is disabled, the print job never reaches the printer and instead sits in the spooler directory, waiting for the printer to become active again. Please note that we don't want to use the `reject` command, because `reject` instructs the print spooler to reject any jobs intended for the printer, so the print job never gets filtered or queued for printing. Figure A shows the difference between the `accept`/`reject` and the `disable`/`enable` commands.

Assuming the printer you're working with is named `hr_printer`, the appropriate `disable` command would be

```
$ disable hr_printer
```

Next, we'll send the print job to the printer. To do this from the command line,

Figure A



The reject command prevents the spooler from accepting a print job, while the disable command only prevents the printer from printing it.

issue an `lp` or `lpr` command. After entering the `lp` command, make a note of the request ID of your print job. For example, here we're going to print the */etc/hosts* file to `hr_printer`:

```
$  lp -dhr_printer /etc/hosts
request id is hr_printer-15 (1 file(s))
```

Thus, our request ID is `hr_printer-15`.

If you're printing from an application instead of the command line, go ahead and submit the print job from within the application. After submitting the print job, change to the temporary spool directory for the printer and list the files in that directory, like this:

```
$ cd /var/spool/lp/tmp/hr_printer
$ ls -al
```

However, when you do so, keep in mind that others may be using the printer as well, so you might not immediately recognize your own print job. You may find it best to select an infrequently used printer at a time when no one else is likely to use it.

At this point, our procedure will vary slightly, depending on the type of print job submitted and the way in which the job was submitted. Depending on the type of printer filter used for the chosen printer, you'll find either the actual filtered file in this directory, or you'll find a log file that tells you the actual print file's location. In our example, because the request ID shown above is `hr_printer-15`, you'll want to look for files in this directory that include the number 15.

PostScript filtered print jobs

First, we'll examine the case of print jobs that have been converted to PostScript format by a print filter. If the file has been filtered, as in the case of a PostScript file, you'll see three files in this directory for your print job, named 15, 15-0, and F15-1. In this case, we want to capture file F15-1. This file will be the PostScript-filtered version of the original file you submitted.

Simply copy this file to another location, cancel your print request, and re-enable the print queue:

```
$ cp F15-1 /tmp
$ cancel hr_printer-15
$ enable hr_printer
```

The file */tmp/F15-1* is the print job you wanted to capture. You can now review and manipulate this file as desired.

Nonfiltered print jobs

If the submitted file hasn't been filtered, you may need to take an extra step or two to locate the file. In this case, the spool directory may contain only one file, named *15-0*. Looking at this file with the `cat` command, your output may look something like this:

```
$ cat 15-0
C 1
D hr_printer
F /hr_apps/reports/970722
P 20
T print_job_15
t simple
U hr_user1
s 0x0000
v 2
```

The line beginning with the letter F tells you the print file's location. In this case, the spool file's name is */hr_apps/reports/970722*. The spooler didn't copy this file to the spooler directory but simply indicated where to find the file in the file system. The same thing happens when you submit a print job with the `lp` or `lpr` command when you don't specify the `-c` option. (The `-c` option tells `lp` to copy the file before spooling, so you can delete the file without worrying about whether the printer is finished with it. In this case, `lp` goes ahead and copies the file into the spooler directory.)

To solve the problem at hand, we'll copy our HR file to another location, cancel the print job, and re-enable the print queue with the lines

```
$ cp /hr_apps/reports/970722 /tmp
$ cancel hr_printer-15
$ enable hr_printer
```

The file */tmp/970722* is the print job you wanted to capture. Again, you can review and manipulate this file as desired.

Final thoughts

The method we've outlined is a quick-and-dirty way of temporarily capturing print jobs, before the job is sent from the print spooler to the physical printer. If you need more information about your print jobs, or if a site is very busy and you can't disable a print queue as needed, then you must begin investigating more complex techniques, such as modifying the printer interface script, */etc/lp/interfaces/hr_printer*. ❖

Alvin J. Alexander is an independent consultant specializing in UNIX and the Internet. He has worked on UNIX networks to support the Space Shuttle, international clients, and various Internet service providers. In the last three years, he has provided UNIX and Internet training to more than 400 clients.

SYSTEM-INTEGRITY ALERT

Beware the pings of death!

Over the last few months, users have encountered two problems in Solaris that can cause your computer to crash if it receives invalid `ping` commands. Luckily, Sun has a great reputation for getting out patches for problems affecting the security and stability of Solaris, so we'll give you the appropriate patch numbers along with the problem reports.

The loopback ping

Someone recently discovered that a broadcast `ping` to the loopback address causes Solaris to panic. It's quite possible that other broadcasts to the loopback address may cause the same behavior. This bug affects Solaris 2.3 through 2.5.1. (Sun caught the problem early enough to fix it in the newly-released Solaris 2.6.)

Until patches come out for your version, Sun recommends that you tell Solaris to ignore all broadcast `ping`s with the following command:

```
$ /usr/sbin/ndd -set /dev/ip
➥ip_respond_to_echo_broadcast 0
```

You'll probably want to put this command in one of your startup scripts so it will run whenever you boot your computer. Currently, Sun has patches available for the following versions of Solaris:

	SPARC	x86
v2.5	103169-12	103170-12
v2.5.1	103630-09	103631-09

Patches for previous versions are still under development.

Solaris x86—the ping of death

Under Solaris x86, there's an additional problem with `ping`: If you send a `ping` in too large a packet, it can force your system to reboot. However, this problem isn't limited to the `ping` command—if your machine receives *any* IP packet larger than the maximum size (64KB), then your machine may crash.

This problem affects only Solaris x86 versions 2.4 through 2.5.1. Patches have been available for Solaris x86 2.5 and 2.5.1 for a while, so if you've installed a Jumbo Kernel Patch recently, you've probably already repaired this problem. If not, the patches listed in the previous section repair this problem as well.

Summary

If you're connected to the Internet, or have mischievous users on your internal network, you should definitely get the appropriate patches ASAP. While the `ping` problem isn't a common one, as the word spreads, perverse users may decide they have to attempt it on your server. You can save yourself some headaches if you're prepared.

You can find the patches we referred to in this article at *ftp://sunsolve1.sun.com/pub/patches*. Because of the lag time involved in publishing this journal, you may find that patches are now available for other versions of Solaris or that some even newer patch files are online. Be sure to check the ftp site for the latest patches. ❖

SYSTEM ADMINISTRATION

Automating file system backups on Solaris 2.x

By Jerry L. M. Phillips, M. S.

An age-old question familiar to real estate agents goes something like this: "What are the three most important factors in real estate? Location, location, and location!" A similar statement applies to system administration, i.e., "What are the three most important tasks of a system administrator? Backup, backup, and backup!" In this article, we'll explore a backup procedure using `mt` and `ufsdump`, then we'll create a `cron` job that executes our backup procedure for us.

Why backup?

Any system administrator with the responsibility of maintaining data integrity on an organization's systems must establish backup procedures in order to protect the organization's investment. A fledgling administrator may not realize the importance of this task. It turns out that you don't rely on backups just to prevent data loss in the event of equipment loss or failure. All too often, the system's users make mistakes and delete critical data—and it's up to you to get it back for them.

Just how often should you back up your file systems? The answer is "It depends." In fact, the frequency depends on how you answer the following questions:

- How critical is your data?
- How often does it change?
- How fast is your backup device?
- How many file systems do you have?
- How large are your file systems?
- How much time can you dedicate to the task?
- How quickly do you need to be able to recover?

You'll have to balance these factors when determining a backup schedule. But you really should think about these questions at the time you're designing your file systems. For example, suppose you have three classes of data. The most important class is the critical and frequently changing data produced by your users, such as sales information and a customer-relations database. Next in importance is data that doesn't change as often, such as internal reports and letters. The rest of your system consists of applications and other executable files.

Obviously, you don't need to back up your applications and executables very often, if ever; on the other hand, you must back up your critical data frequently. (After all, you can always reinstall your applications and executables should you lose the file system.) A common technique is to back up the application and executable file systems only after you install or remove code. You might then choose to back up your critical data daily and your less-valuable data weekly.

As we said, it depends: If the amount of critical data is small and your backup system fast, you might elect to back up your critical data twice a day, at lunch and in the evening, or even between shifts. Because some of your data doesn't change much, you might choose to perform an incremental backup on that data every night, since it'll probably take little space or time.

Start with a command

Now, how do you begin? And what software should you use? Some tape drive manufacturers provide programs to streamline the backup process; and many third-party software vendors would be happy to sell you a good backup program. However, I prefer to use `mt` and `ufsdump` and a few easily managed command-line procedures and execute my file system backups on cartridge tape drives. After all, these easy-to-use commands are distributed with Solaris, so you won't have to worry about whether the manufacturer will go out of business or a new version will force you to change your procedures.

Using the mt command

You can use the `mt` command to send commands to your tape drives. The format of the command line is

```
mt  -f tapename  command...  count
```

where `-f tapename` is an optional parameter that allows you to specify the tape drive you want to work with, `command...` is a list of commands you can send to the tape drive, and `count` specifies the number of times to perform the tape commands.

The list of commands that `mt` supports is shown in Table A. Please note that not all commands accept a count, but those that do reference `count` in their descriptions.

For example, if you want to determine the status of your cartridge tape drive, you can use the following command (depending upon your hardware configuration):

Table A: *The mt command supports a complete set of commands for tape management.*

Name	Description
`eof, weof`	Write `count` End Of File (EOF) markers to the end of the tape
`fsf`	Skip past `count` EOF markers
`fsr`	Skip forward `count` records
`bsf`	Skip backwards `count` EOF markers
`bsr`	Skip backwards `count` records
`nbsf`	Skip backwards `count` files
`asf`	Absolute position to file `count`
`eom`	Skip past the last EOF on tape
`rewind`	Move to the Beginning Of Tape (BOT) marker
`offline, rewoffl`	Rewind the tape and take the tape unit off-line (if applicable)
`status`	Print the status of the tape unit
`retension`	Move to the end of the tape and rewind so the tape has uniform tension
`erase`	Erase all the data on the tape

```
# mt -f /dev/rmt/0n status
Archive Python 4mm Helical Scan tape drive:
Sense Key(0x6)= Unit Attention residual= 0
retries= 0
file no= 0 block no= 0
```

While you can use the `mt` command for many purposes, we'll concentrate on using it to rewind the tape. We do so to be sure we start at the beginning of the tape, so we'll overwrite old backups. You can rewind your tape with a command like:

```
# mt -f /dev/rmt/0n rewind
```

Please note that, not all tape devices support the entire set of commands. Some tape drives, for example, don't support the retension command:

```
# mt -f /dev/rmt/0n retension
/dev/rmt/0n retension 1 failed: Inappropriate
ioctl for device
```

Quick Tip: The `mt` command defaults to using the tape drive */dev/rmt/0*. However, this device automatically rewinds, so we've been using the `-f /dev/rmt/0n` parameter to specify the device for tape drive 0 that doesn't rewind. But, you don't have to specify the `-f /dev/rmt/0n` parameter each time. The `mt` command will use the `TAPE` environment variable, if available, to specify the tape device to use. Thus, you could define the `TAPE` environment variable, then use the `mt` command with much less typing:

```
# TAPE=/dev/rmt/0n; export TAPE
# mt retension
# mt asf 5
```

Using the ufsdump command

You may currently be using `tar` or `cpio` to do your backups, but there are good reasons to use `ufsdump` instead. `ufsdump` allows you to perform simple incremental backups, and it offers greater control over the tape drive parameters.

When you use the `ufsdump` command to back up a file system, you must ensure that the file system is inactive. If the file system changes, your backup tape will be a combination of "before" and "after" and will probably be useless. Thus, be sure to `umount` each file system before you back it up; you can also put the system into single-user mode first. The syntax of the `ufsdump` command is

```
ufsdump options  arguments  files
```

where *options* is a single string of one-character `ufsdump` command options, and *arguments* are strings associated with the options, specified *in the same order* as the command options you choose. For example, you could use the following command to back up a slice of your disk drive:

```
# /usr/sbin/ufsdump 0bcdsfu 126 28633 3125
➥ /dev/rmt/0n /dev/dsk/c0t3d0s0
```

At first glance, the command line appears quite intimidating. However, once you deconstruct the command line into its individual components, it's not so bad. Let's take a look at the option string (`0bcdsfu`) first.

The initial character, `0`, is the dump level, which tells `ufsdump` the specific files to back up. There are 10 dump levels (`0` through `9`); `0` directs `ufsdump` to back up every file in the entire file system. (Please see the sidebar "Dump Levels and `ufsdump`.")

Next comes the `b` command, which specifies the number of blocks to back up per write operation. For cartridge tape drives, the default blocking factor is 126. Please note that this is the first command that uses an argument, so the first argument string, `126`, refers to the blocking factor we select.

Next comes the `c` command, which tells `ufsdump` that we're using a cartridge tape drive. The `d` command, specifying the tape density in bits per inch, follows. Again, the `d` command accepts an argument, so the next argument string, `28633`, specifies the number of bits per inch for the `d` command. Then comes the `s` command, which tells `ufsdump` the number of feet in each cartridge—the argument `3125` tells `ufsdump` that there are 3,125 feet in each tape cartridge.

The `f` command option, like the `-f` option used by `mt`, specifies the name of the tape drive device. By default, `ufsdump` uses the */dev/rmt/0* device, which rewinds the tape when the command ends—a problem if you're planning to back up multiple file systems to a single tape. So we set our next argument string to */dev/rmt/0n*, telling `ufsdump` to use the tape driver that doesn't rewind the tape. Finally, the `u` command tells `ufsdump` to update the backup record in the file */etc/dumpdates*.

Now that we've processed the command option string and used the command arguments, the next (and final) entry on the command line is */dev/dsk/c0t3d0s0*. This command specifies the file system we want to back up.

Quick Tip: Earlier, we showed you how the `mt` command uses the `TAPE` environment variable to specify the tape drive. The `ufsdump` program, however, ignores it. You can still use the `TAPE` environment variable with `ufsdump` by using it as the argument string of the `f` command option, like so:

```
# ufsdump 0cf $TAPE /dev/dsk/c0t5d0s3
```

Automating the process

What if you must back up several partitions on one or more disk drives? One thing you don't want to do is manually type in a `ufsdump` command for every partition on every hard drive you want to back up, waiting for each backup to finish before entering each command. Instead, you should build a script file containing the appropriate combination of `mt` and `ufsdump` commands.

As an example, Figure A contains a sample script named *BU.riker*, which I wrote for backing up my personal workstation. Please note that I placed the script in my root directory, as the script `mounts` and `umounts` various file systems to ensure that they're not in use.

If you don't rewind the tape, you may output more than one backup onto the same

Figure A

```
#! /bin/sh

# ufsdump routine to backup DNS disk drives

# Set working directory and TAPE variable
cd /usr/bin
TAPE=/dev/rmt/0n

# rewind cartridge tape to beginning of tape mt rewind

# umount my data file systems
umount /usr/data97 /usr/data96 /usr/data95      \
       /usr/data/94 /usr/data/93 /usr/data/92   \
       /usr/data/91

# back up the file systems, most recent first
ufsdump 0cf $TAPE /dev/dsk/c1t3d0s1
ufsdump 0cf $TAPE /dev/dsk/c1t2d0s3
ufsdump 0cf $TAPE /dev/dsk/c1t2d0s4
ufsdump 0cf $TAPE /dev/dsk/c1t2d0s7
ufsdump 0cf $TAPE /dev/dsk/c0t3d0s5
ufsdump 0cf $TAPE /dev/dsk/c0t3d0s4
ufsdump 0cf $TAPE /dev/dsk/c0t3d0s3

# rewind tape cartridge to beginning of tape
mt rewind
```

This script backs up several critical file systems on my workstation.

tape, which could result in you having to intervene manually during the backup process in order to switch tapes. I prefer writing only one backup to the cartridge tape, which allows the backup process to run unattended. However, in situations involving large disk drives, switching tapes is unavoidable.

Back up like clockwork

Turning the backup process into a script is only the first step. Once you get your script working, it's not necessary to execute it manually. Now that we have our backup script, let's add it to the `crontab` for the superuser to ensure that it runs only Sunday through Thursday nights. (We covered the `crontab` command in the article "Scheduling a Job for Periodic Execution" in the October '96 issue and followed up with a few `crontab` tricks in the article "The Second Sunday of Each Month…" in the September '97 issue.)

To add the script, you must get the current `crontab` for the superuser, add an entry for our backup, then submit the new `crontab`. We can get a copy of the `crontab` into the file *X* by executing the command

```
# crontab -l >X
```

Now, just invoke your favorite text editor, and add the appropriate entry for your backup script. Just so you don't have to look it up, the format of a `crontab` entry is

```
m h d m w command
```

where *m*, *h*, *d*, and *m* specify the minutes, hours, days, and months to execute your command, and *w* specifies the weekdays. If you don't want to specify a field, use an asterisk in its place to tell `cron` to ignore the field. Since we'd prefer to execute our backup script at 2:00 a.m. each Monday through Friday, we'll add the entry

```
0 2 * * 1-5 /usr/local/scripts/BU.riker
```

Now we simply submit the command table back to `crontab` with the command

```
# crontab <X
```

Conclusion

Using a script to drive `mt` and `ufsdump` allows you to automate just about everything in your backup process, except for having to switch tapes. When you write your own backup script, you might want to have a different script for each night of the week, add error checking, and customize other scripts to suit your environment. And finally, you should spend some time reading the `man` pages on the `mt`, `ufsdump`, `cron`, and `crontab` utilities. ❖

Jerry L. M. Phillips, M.S., is director of the Academic Computer Center at Eastern Virgina Medical School. Besides his administrative duties, he manages Sun/Solaris-based platforms for the medical school, including DNS, WWW, anonymous FTP, mail, and Usenet News servers.

MANAGEMENT

Using postprint to print plain ASCII files to your PostScript printer

By Al Alexander

When working with plain text documents (such as program listings or data dumps), you may want to print more than the standard number of lines per page, change the standard font, or modify the standard column width of the printed file. If you have Solaris 2.x and a PostScript-capable printer, the solution is simple: Solaris 2.x comes with a handy print filter named `postprint`, which can convert ASCII documents into PostScript format. Using the `postprint` command, you can control many of the document-formatting parameters, such as number of lines per page, page orientation, font, font size, and magnification.

The scenario

For example, I recently worked with a human resources (HR) group that was having problems with an HR software package. Because of some limitations in the software, HR couldn't print a report 120 columns wide in landscape mode, even though they could view the report in that mode on their computer screens. If this were a one- or two-page report, there may have been other workarounds. But this company had over 2,000 employees, and this report was being printed each month—containing one record for each employee. This report quickly became a big problem.

The first part of the problem entails capturing the output from the HR software before the report is sent to the printer. If the information is currently printing to your screen, you could use the `tee` filter to capture it to a file so you could print it later. If your application is currently printing the data, see the article "Capturing Print Jobs Before They're Printed," in this month's issue, for one possible approach. Once you have the data you want to print, the second part of the problem is to convert the document from ASCII to PostScript format.

Let's assume that you've captured the plain-text document, and you've saved it as */tmp/doc_ascii*. Using the `postprint` command, you can easily convert the document to PostScript format. In fact, the hardest part of the process will be deciding how you want the output to look.

After a few iterations of reading the `man` page for `postprint` and trying out some test prints to our PostScript printer, we found that we could see all 120 columns of our report by changing the paper orientation to landscape and printing the document in 6-point Courier. We did both by entering the following commands:

```
$ cd /tmp
$ /usr/lib/lp/postscript/postprint -fCourier
➥ -s6 -plandscape < doc_ascii > doc_ps
```

In this example, `postprint` reads its input from the *doc_ascii* file, performs the appropriate conversions, and writes its output to the *doc_ps* file. The `-fCourier` option tells `postprint` to use the Courier font, `-s6` specifies six-point type, and the `-plandscape` option changes the print orientation to landscape mode.

Because a PostScript file is really just a text file (with embedded PostScript commands), you can view this new file with the `more` command (or `pg` or the `vi` editor) by typing

```
$ more doc_ps
```

Initially, everything you see will be PostScript printing commands. After scrolling through enough of the output, you'll eventually see your original data at the end of the file.

The formatting options you applied to the file will be displayed at the beginning of the resulting PostScript file. The first few lines of the PostScript file for our example are shown in Figure A.

Figure A

```
%!PS-Adobe-2.0
%%Version: 3.15
%%DocumentFonts: (atend)
%%Pages: (atend)
%%EndComments
/font /Courier def
/pointsize 6 def
/landscape true def
% @(#)postprint.ps      1.26 09 Dec 1991
%ident  "@(#)lp:filter/postscript/postscript/postprint.ps   1.1"
```

These are the first few lines of the new PostScript file, ps_doc.

Now that you have your PostScript file, you can send it to your PostScript printer. For example, if your PostScript printer was named *hp_postscript*, you'd print it like this:

```
$ lp -dhp_postscript doc_ps
```

The PostScript commands embedded in your file and your PostScript printer take care of all the hard work.

Another advantage to using the `postprint` command is that it comes with Solaris. However, other solutions are available. Solaris also has the `mp` command, while Adobe offers Enscript, and others are available over the Internet. The `postprint` command solved the problem for us, but you might want to investigate these other solutions if your needs are more demanding.

Conclusion

If you have access to a PostScript printer, the `postprint` command is a great utility for easily converting ASCII documents into PostScript format. Using PostScript formatting options, you can easily modify the font, font size, print orientation, number of lines per page, and other text-formatting options. ❖

I'd love to install it, but there's no room!

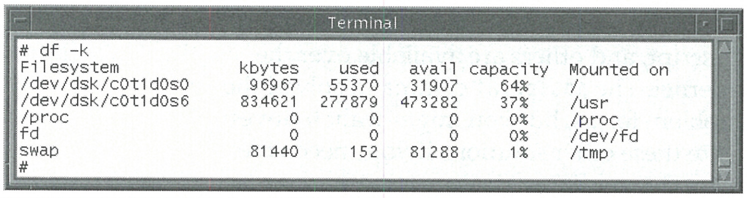
It's bound to happen sooner or later. You need to install an application, but there's no room for it on the recommended file system. Sure, you could rearrange your file system by backing it up and restoring it to a larger slice on another hard drive. But what if a larger slice on your hard drive isn't available? You could rearrange the file systems on your computer, but that can take awhile. Your boss/user/client/team wants to use the application *now!*

There's no need to panic. Many packages provide instructions on how to install them in another location. Unfortunately, many don't! Often, you can trick an application to make it install in another file system, allowing it to think that it installed in the desired location. In this article, we'll show you three tricks you can use to install a package to an alternate location.

Installing a package

We recently received a copy of the Java WorkShop that we needed to install so we could start developing Java applets for a Web page. The Java WorkShop is a small package that is normally installed in */opt*. However, our test machine doesn't have a separate */opt* file system, and our / file system has little space available, as shown in Figure A. (Although it has enough space to install the Java WorkShop, we'd be cutting it closer than we'd like.)

Figure A



```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t1d0s0      96967   55370   31907    64%    /
/dev/dsk/c0t1d0s6     834621  277879  473282    37%    /usr
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
swap                   81440     152   81288     1%    /tmp
#
```

Our test machine has limited resources and no /opt partition.

Normally, you'd install a package using the command

```
# pkgadd -d directory pkgname
```

where `directory` specifies the location of the package to install and `pkgname` specifies the name of the package you want to install. Then you simply answer any questions, and `pkgadd` installs the package for you.

Selecting a new root directory

One way to put a package into another file system is to tell the `pkgadd` command to treat the other file system as the root file system, which you can do by giving the `pkgadd` command the `-R directory` option. Please note that if you do so, you must ensure that the directory structure mimics whatever the package is expecting. For example, the Java WorkShop defaults to installing itself into the */opt* directory. If we want to install it to our */usr* file system, we need to create the */usr/opt* directory before adding the package, like this:

```
# mkdir /usr/opt
# pkgadd -R /usr -d . SUNWjws
```

Using this command, we added the package to our */usr* file system, in the directory */usr/opt/SUNWjws*.

If you use this technique, however, you should be aware of two things. First, the installation process will create a new directory tree in the file system where you installed your program—the */var* directory tree. This happens because Solaris keeps a database of the packages you install so it can remove them later with the `pkgrm` command, if necessary.

Solaris stores the information about the package in the */var* directory tree. Since the -R *directory* technique tells the `pkgadd` command to treat the directory as the root, the `pkgadd` command updates the installed package database in your new file system—hence the new */usr/var* directory on our test system.

This side effect has both bad and good results. The bad side is that your installed package database now resides in two locations and is now less useful. If we want to see the information on the Java WorkShop package, running the `pkginfo` command would yield

```
# pkginfo SUNWjws
ERROR: Information for "SUNWjws" was not found
```

because the default-installed package database is located in */var*, not */usr/var*. To see the information, we must remember the file system in which we installed the package and remember to specify the same -R option to all the package commands we want to use for that package.

There's a worse problem, however: Since there are now two (or possibly more, if you use this technique to install other packages on other file systems) installed package databases, you run the risk of corrupting the system. To see why, consider this: Suppose you use this technique to install a package to an alternate location, and it replaces a standard system file. Later, you install a patch, and it updates the same system file. If you then use `pkgrm` to remove the package, it will restore the original system file rather than leaving the patched version in place. At the least, this could deactivate the patch, and at worst, corrupt the oper-ation of your system.

The good side to this technique is that if you're developing packages for use on other systems, you can use this technique to test a package installation without damaging your working environment. (This statement holds true as long as your package doesn't replace any of your system files.)

Changing the defaults for the package commands

Another way you can install a package to a different location is to modify the defaults to the package commands. These defaults are specified in a file named */var/sadm/install/admin/defaults*. When you install Solaris, the default values are set as shown in Figure B.

Figure B

```
#ident  "@(#)default    1.4     92/12/23 SMI"
➥/* SVr4.0  1.5.2.1     */
mail=
instance=unique
partial=ask
runlevel=ask
idepend=ask
rdepend=ask
space=ask
setuid=ask
conflict=ask
action=ask
basedir=default
```

When you install Solaris, it sets these default values for the package commands.

Notice the line in blue: This line specifies the action for how you decide the base directory in which to install a package, and it's currently set to `default`. What you really want to do is copy this file to another location and change the line to read

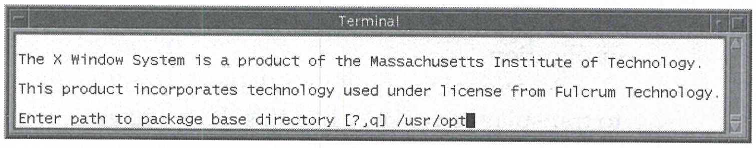
```
basedir=ask
```

If you want to install a package using this new *default* file, you simply specify the name of the *default* file with the `-a` option, like this:

```
# pkgadd -a /home/marco/default -d . SUNWjws
```

Now, when you install the package, the `pkgadd` command will ask you for the directory in which you want to install the package, as shown in Figure C.

Figure C



```
The X Window System is a product of the Massachusetts Institute of Technology.
This product incorporates technology used under license from Fulcrum Technology.
Enter path to package base directory [?,q] /usr/opt
```

After you override the system defaults, `pkgadd` will ask you where you want to install the package.

This technique is great for shops that want to manage applications and disk space by department. You can give each department its own file system on which to install its applications, and it's easy to tell how much space each department is using for applications. Thus, when a depart-ment's file system is full, you know whose budget to debit for new hard drive space.

Use a symbolic link

Perhaps the simplest and most well-known method to install a package when a file system is full is the symbolic link method. In this method, you simply create a directory on the file system on which you want to install the application. Then you create a symbolic link in the */opt* directory that points to the directory you just created and install the package normally. So, using the same example, if you want to install the Java WorkShop to the directory */usr/opt/SUNWjws*, you could execute these commands:

```
# mkdir -p /usr/opt/SUNWjws
# ln -s /usr/opt/SUNWjws /opt/SUNWjws
# pkgadd -d . SUNWjws
```

The advantage of this technique is that it's quick and easy. If you find that you need to move a package from one file system to another, you can use the same trick. For example, suppose you want to move a smaller package from the */opt* file system to another file system to make room for a larger package. You first copy the entire directory tree for the package to the new file system, then rename the package directory holding the original to another name. Now you can create your symbolic link to the alternate location for your package:

```
# mkdir -p /usr/opt/PACKage
# cd /opt
# tar cf - PACKage | (cd /usr/opt; tar xf -)
# mv PACKage PACKage.backup
```

You're ready to test your application at its new location. If all goes well, you can remove the directory and prepare to install your new application.

Occasionally, you'll find that you don't want to transplant a package like this. If you have many shells and you're working with symbolic links, you can easily find yourself in the wrong subdirectory. For example, in the code snippet below, we're using the Bourne shell, and we begin in the */opt* directory:

```
# pwd
/opt
# ls
PACKage   SUNWaadm  SUNWabe
SUNWaman  SUNWits   PACKage
# cd PACKage
# cd ../SUNWabe
../SUNWabe: does not exist
# pwd
/usr/opt/PACKage
```

Surprise! We couldn't get to */opt/SUNWabe* because we're in */usr/opt/PACKage*, and there's no */usr/opt/SUNWabe* directory.

Most applications use files from their package directory. If the application uses many of these files very frequently, you might notice a degradation in performance because symbolic links incur a bit of overhead each time they're looked up. Specifically, they force Solaris to perform an extra disk lookup. This slowdown should occur rarely enough that it shouldn't be a primary concern; however, if you notice your system losing speed after you move a package using this technique, the situation may be worth investigating.

Please note that other factors are more likely to cause the slowdown in your system's performance. Moving a frequently used application to a file system on a drive that's already used heavily will affect all applications on that drive. If the drive is a frequently used swap surface, then every application may take a performance hit.

Conclusion

In UNIX, there are often many ways to accomplish a task, each with its own tradeoffs. In this article, we've explored some of the tradeoffs you'll see when you need to install a package and you don't have enough room on your target directory. If you're developing and testing packages, specifying another directory as the root file system can be a handy trick. If you're managing packages for different groups and want to give each group its own area, then overriding the defaults during installation is the way to go. However, if you just want to quickly install a package, you'll probably want to opt for the symbolic link approach. ❖

DEVELOPER NEWS

Good news for application developers!

Have you ever wanted to write applications that you could run under both Solaris and Windows? Well, it turns out that Willows Software (a division of the Canopy Group) has made it easier to do so, with its TWIN API product. To make things even better, Willows has placed it under the Gnu General Public License (GPL), so you can download the tool, source code and all, from the Internet—and pay no licensing fees.

What is the TWIN API?

Now you're probably asking yourself, "How can the TWIN API help me write programs that I can run on Windows and Solaris?" It turns out that the TWIN API provides libraries with a Windows-compatible Application Programming Interface (API) for both 16-bit (i.e., Windows 3.1) and 32-bit (Windows 95 and Windows NT) compatibility, the Registry API, the WinSock API, the CommonDialog API, etc.

If you write your programs to the TWIN API, you'll be able to compile them with Visual C++ to create Windows programs. The TWIN API package also provides the source code to libraries that will internally convert the Windows API to the appropriate operations under Solaris. In addition, it provides for file system operations, windowing operations, etc. Obviously, the TWIN API can't do everything, so in order to get the best results, you'll have to avoid a few problem areas. For example, you don't want to use any special features in your compiler, or you'll limit the portability of your application. The rest of these tips are documented on the Willows Software Web page.

An added bonus

The TWIN API also provides an Application Binary Interface (ABI), so your Windows programs can run on other systems, such as the Apple Macintosh, Caldera OpenLinux, and Novell NetWare. Since the ABI source code is available, no doubt some Solaris x86 developer is already porting the TWIN ABI to Solaris x86 to allow Solaris x86 users to run Windows programs!

Why this generosity?

Ralph Yarro of The Canopy Group Inc., says that "APIs should belong to the community at large; no one entity should control them. We want users to have the freedom to choose the platforms that best serve them and not be limited by the lack of applications that run on them. That's why we support not only this action, but other developments like the Wine Project and Java as well. We greatly appreciate the investment of time and money that developers have made to the Windows API. It is our intention to give them a free and efficient alternative for moving their products to other platforms without further investment."

Rob Farnum, the inventor of the technology, continues to work on the TWIN API under contract with The Canopy Group, coordinating the efforts of developers on the Internet to make the Windows API ubiquitous throughout the UNIX world. Hopefully, this action will broaden the application market for UNIX, allowing UNIX more market share of those corporate desktops.

Where to get it

If we've piqued your interest, you should visit the Willow Software site at *www.willow.com* to get more information. You can download the source code from its FTP site at *ftp.willows.com*. Willows Software has also recently created two mailing lists for the product, one for developers (twindev@willows.com) and one for users (twinusers@willows.com). If you'd like to subscribe to either mailing list, send E-mail to the address(es) you're interested in, with the word *subscribe* as the subject line. ❖

SYSTEM ADMINISTRATION

Dump levels, ufsdump, and tape rotation

The `ufsdump` program uses the concept of *dump levels* to allow you to make the process of backing up your file systems as efficient as possible. The dump level tells `ufsdump` which files it must back up. Each time you back up your file systems with `ufsdump`, it records when you backed up the system and at which dump level. The `ufsdump` program uses this data to decide which files are candidates for backing up the next time you run it.

How it works

The algorithm is simple: Every time you run `ufsdump`, it looks at the date of the last backup you did with a lower dump level. Then `ufsdump` backs up each file that changed since that date. Since there's no dump level lower than 0, dump level 0 always backs up every file on the file system.

Since backing up fewer files takes less time, you can use higher dump levels to make quick backups of the files that have changed over a short period of time. The tradeoff is that you can't restore a file system from one of these incremental backups. You must have a full system dump, the last incremental backup, as well as the last incremental backup of each lower level that exists in order to restore your files.

Restoring from a backup

The only problem with using multiple backup levels is that it takes longer to restore your data if you lose the contents of a file system. The reason is that in order to restore your data, you must first restore the last level 0 backup that you did. Then, you must restore the last level 1 backup, the last level 2 backup, etc. However, you should be restoring your data much less frequently (hopefully never!) than you back it up, so you save time. Thus, you must make a tradeoff between the amount of time you want to spend each month backing up your files vs. the amount of time it would take you to recover the data.

SunSoft Technical Support
(800) 786-7638

PERIODICALS MAIL

Please include account number from label with any correspondence.

Tape rotation

By carefully scheduling the dump levels you use, you can have the security of a complete system backup, without spending all the time required to perform a complete system backup every night. For example, in Figure A, we show a backup schedule where every four weeks, you perform a complete system backup, each intervening week gets a slightly higher level backup, each Wednesday gets a still-higher level backup, and each remaining day gets the highest level. The cycle then repeats.

Figure A

Sun	Mon	Tue	Wed	Thu	Fri	Sat
0	6	6	4	6	6	6
2	6	6	4	6	6	6
2	6	6	4	6	6	6
2	6	6	4	6	6	6
0	>	>	>			

A backup schedule like this minimizes the time you spend backing up your system.

So on our first Sunday we build a complete backup, then a level 6 incremental backup on Monday, then on Tuesday, which supersedes Monday's backup. On Wednesday, we do a level 4 incremental backup, which obsoletes the Tuesday tape. The Thursday tape is obsolete after we do Friday's tape, which in turn is superseded by Saturday's tape. The next Sunday, we do a level 2 incremental backup, which obsoletes the Wednesday and Saturday incremental backups. Finally, on our last Sunday in Figure A, we obsolete all previous backups.

Tape management

Since tapes are a somewhat expensive resource, you must manage their use accordingly. However, your data is an even more expensive resource than your tapes—so be sure to keep at least one complete backup of the system in addition to your latest backup, just in case your latest backup turns out to be defective.

You must decide the relative worth of your data vs. the cost of the tapes to decide just how many copies of the system you want to keep. Don't forget to manage your backups carefully, as there can be quite a few tapes to keep track of.

For example, suppose you're following the schedule shown in Figure A, and a full backup of your system takes 15 tapes, the weekly incremental backup uses eight, the Wednesday incremental backup takes five, and each daily incremental backup uses two tapes. Therefore, you must have up to 30 tapes to completely restore the system. A complete backup of the system will require another 30 tapes. Since you'll also want some spares, it can be tedious to track your tapes.

Label each tape carefully, and track their use in log so you can tell what each tape contains. This helps you keep track of which tapes are which. Also, if a user wants a specific file and knows the day it was last modified, you can quickly locate the appropriate tape to use.

Wear and tear

As you use your tapes, be aware that tapes wear out, and tape drives need cleaning. Since tapes wear out, you may want to track how each tape is used, so you know when to replace it. You don't want to wait for a backup tape to fail before replacing it, since it could fail during the restore operation. Decide how many times a tape should be used before you replace it. Given a clean tape drive and careful use, tapes should last a reasonably long time before you should replace them.

Keeping your tape drive clean, is one way to prolong the life of your tapes. Yes, it increases wear on the drive. However, considering how many tapes you'll have, the cost of the tapes may exceed the cost of the drive!

Be flexible

Don't be shy about adapting these techniques to fit your particular situation. When you plan your backup strategy, be sure to leave some holes in your calendar. This way, should an emergency arise, you can dispatch it, then slip your backup schedule a bit. Having these holes in your schedule can keep the schedule slippage to a minimum. ❖

Printed in the USA
This journal is printed on recyclable paper.