March 1997 • $11.50
VOL. 3 NO. 3

INSIDE SOLARIS
Tips & techniques for users of SunSoft Solaris™

in this issue

Visit our Web site at http://www.cobb.com/sun/

Detecting inactive accounts on your network in an NIS+ environment

By Gaetan Boudreau

For security and maintenance purposes, it's always a good idea to find and delete inactive accounts. When an employee leaves the company, finding and destroying the employee's accounts on all computers is relatively simple.

However, networks with many machines may still contain inactive accounts for the employed users who no longer use that computer for serious work. Every account—especially an unused account—on a system is a weak point that may be attacked by an unscrupulous person. On these accounts, there's no user to complain about misplaced files, inappropriate billing of services, etc.

In this article, we're going to show you a simple way to detect inactive accounts on a system where you use NIS+ and the automount service for user accounts.

Put an inactivity timeout on the accounts

It's possible to configure user accounts with a maximum inactive period. If a user leaves an account inactive for more than the specified number of days, the account is automatically locked, and the user can no longer access it. This prevents a hacker from gaining access to the account after this window has passed.

The problem...

Unfortunately, this solution isn't bulletproof. As you can see on page 166 of the *Solaris 2.5 NIS+ and FNS Administration Guide AnswerBooks*, "this feature is tracked on a machine-by-machine basis, not a network-wide basis" using the information stored in each system's */var/adm/utmp* file. (This AnswerBook is supplied in the System Administrator's AnswerBook package. If you have a Solaris server and haven't installed it, see the article "Don't Forget the Rest of the AnswerBooks" on page 10.)

This is a problem for user accounts in an NIS+ multiserver environment. Suppose, for example, that we set the maximum inactivity timeout for user accounts to 30 days. A user logs in to host A and host B at the beginning of the month, then every day to host A. The next month he logs back to host B but finds that he's locked out of that host because of the inactivity timeout, even though he used the network every day through other machines. This is a case when you don't want to

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@merlin.cobb.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: cr@merlin.cobb.zd.com.

Staff
Editor-in-Chief ..... Marco C. Mason
Contributing Editors ..... Gaetan Boudreau
Production Artists ..... Margueriete Winburn
Natalie Strange
Editor ..... Karen S. Shields
Publications Coordinator ..... Linda Recktenwald
Marketing Coordinator ..... Marcella Able
Circulation Manager ..... Mike Schroeder
Editorial Director ..... Linda Baughman
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

lock out the user, but it happens anyway.

You can also have a different problem: You want to lock out the user but can't. Since it's only at login time that the program compares today's date with the last login on that host, it doesn't work when the user has never logged in on the host. So if you have many machines on the network, the user locked out of one could always get in from a machine he never used before. Thus, a locked, inactive account on one machine doesn't prohibit entry from another machine.

Figure A

```
$
total 14
lrwxrwxrwx  1 root  nobody    16 Dec 31 07:23 newDir
dr-xr-xr-x  2 root  sys     6144 Sep 26  1995 oldDir
```

A new file or directory prints the time after the month and day, where an old one prints the year.

A missing feature...

One more thing that's not so great about this feature is that even if you have local users using a single server, you won't be notified of a user going over his inactive period and being locked. You could run a script giving you all the locked (*LK*) accounts, but you won't get just the most recent ones. The usual case is where the employee leaves the company without anyone alerting you. The ex-employee's data and account will sit there, and the account will never be locked since locking only happens at login time.

Our solution

Our users are in a NIS+ multiserver environment. We use the automounter for the home directories of the users. They can access their accounts by `login` from any server in terminal mode. They can also access their accounts from their PCs with Sun's PC-NFS or PC-NFSpro software, mounting the network drive as a DOS drive with NFS Home after being authenticated with `rpc.pcnfsd`.

We've learned that the date of the home directory changes to the current date each time the user logs in from any location. Using this information, you can easily write a simple script to locate the inactive accounts.

You must write a script that lists the home directories in chronological order, as with the `ls -lt` command. (Use `ls -lrt` if you want them in reverse order). Then you can use the output of this command to locate users who haven't logged in recently enough.

One way you can simplify the script is to use a maximum inactivity period of six months. Why six months? It turns out that the `ls -l` command writes the date for a file younger than six months in a different format than it does for an older file. For example, Figure A contains a directory listing containing two directories, one older and one younger than six months.

As you can see, a new file or directory entry prints the time after the month and day, whereas an older directory entry prints the year instead. So the trick is to list the directory containing the home directories of your user accounts and search for only the ones that print the year instead of the time. Thus, a simple command to find an account that's been inactive for over six months is

```
ls -lt UsrHomeRoot | grep -v :
```

Here, we assume that your user directories are mounted at the *UsrHomeRoot* directory, which would be something like */export/home*. We use the `ls -lt` command to list all the directories in your *UserHomeRoot* directory and the `grep -v :` command to print all files that *don't* contain a :, which is used in the time field. Thus, all we have left are entries where the year is present—those directories older than six months.

Obviously, this is only a starting point. You can be a little fancier by executing this script via the `cron` daemon weekly and mail yourself the results. Then you can quickly lock out any accounts that require it. (For information about running a job with `cron`, see the article "Scheduling a Job for Periodic Execution" in the October 1996 issue.)

If you really want to get fancy, you can make a more complicated script that will use this input and automatically lock the accounts with the `passwd -l` *`User`* command, where *`User`* corresponds to the name of the user's home directory. A first attempt at such a script appears in Figure B.

Figure B

```
#!/bin/ksh

# Create a temporary file name to use
# for our user lock script.
REPORT=/tmp/ISOL_LockUser_${$}

# Make sure that the file doesn't exist
rm $REPORT

# Build the user lock script by putting
# a "passwd -l <USER>" entry for each
# user whose home dir is 6 mos old...
ls -lt /export/home | grep -v : | awk '/^d/
  ➥{print "passwd -l " $9}' >$REPORT

# Execute the user lock script
REPORT

# Mail it to root...
mail root <$REPORT

# Delete it
rm $REPORT
```

This simple script will automatically lock out each user who hasn't logged in for six months.

This script first decides on a name for a temporary file, then deletes it from the */tmp* directory, just in case it already exists. Then it scans through the */export/home* directory (the home root directory for all users on our test system) for all directories over six months old, using the tricks we've already discussed. Next it uses `awk` to print out the `passwd -l` *`User`* command, substituting each directory name for *`User`*.

You can use this script as a starting point for your own account-locking script. One thing you'll probably want to do is shorten the activity timeout period, since six months may be too long for your application. Another thing you may want to do is to filter out the directories that aren't user home directories. We didn't bother, since the `passwd` command ignores those entries. Filtering them out will make your reports smaller and clearer, though.

Beware!

Please note that since the inactivity timeout relies on the information found in */var/adm/utmp*, be aware of the ramifications of using the log file trimming techniques we described in the article "Trimming Your Log Files" in the September '96 issue. In that issue, we simply clear the file, removing all information. Instead, you'll want to remove only those entries older than your longest inactivity time, or inactivity detection will fail on your system.

If you're using PCs on your network, and your TCP/IP and/or NFS suite for the PC doesn't change the date of the home directory, you could create a batch file that would execute a remote shell and `touch` the home directory. Figure C shows the one we use.

Figure C

```
rsh hostname touch ~
net login
```

This DOS batch file touches the home directory to ensure that its date stamp is correct for our inactivity detector.

If you're using Samba on your computer, you can use the `PREEXEC` option to execute a script to update the home directory, if that's required.

Conclusion

If you can make it difficult enough, a hacker won't get into your system. However, that could mean it's difficult for you and your users as well. We hope that by using these techniques, you'll plug some security holes and make a hacker's life harder, without adversely impacting you or your users. ❖

Gaeton Boudreau (Gaetan_Boudreau@IRCM.UMontreal.CA) has been a network administrator for nine years at the Clinical Research Institute of Montreal. He manages a mix of SUN, Macintosh, and PC workstations. He received a bachelor's degree in Electrical Engineering from Ecole Polytechnique of the University of Montreal.

Sharing files with Windows computers using Samba

We recently installed Windows 95 on a secretary's desk so she could run Word. At that time we also installed a network card, so we could plug her computer into the network. While we were installing the network, we were pleased to notice that Windows 95 offered us the option to install the SunSoft PC-NFS protocol. Imagine our disappointment when we found that the drivers weren't included with Windows.

Well, we needed the ability to share some text files between the secretary's computer and our network, and we needed it now. Luckily, one of us remembered a package available on the Internet called Samba. With Samba, you could supposedly share printers and directories between UNIX and Windows computers.

In this article, we'll share our experiences with Samba, showing you how to compile, install, configure, and use it. The long and short of it is that it works and works well.

Andrew Tridgell
E-mail: samba-bugs@anu.edu.au
3 Ballow Crescent
Macgregor, A.C.T.
2615 Australia

What is Samba?

UNIX shares file systems over a network using the NFS protocol. When Microsoft added networking to Windows, it created a protocol called SMB to allow Windows computers to share files and printers.

Andrew Tridgell at the University of Australia wrote Samba to allow UNIX to understand and communicate with the SMB protocol. Samba allows you to share both file systems and printers with Windows computers. With it, your Solaris computer may use files and printers on Windows machines, and Windows machines may access files and printers on your Solaris computer.

While Tridgell doesn't charge for Samba, he does state in his documentation:

> "If you want to contribute to the development of the software, then please join the mailing list. I accept patches (preferably in `diff -u` format) and am always glad to receive feedback or suggestions.
>
> You could also send hardware/software/money/jewelry or pizza vouchers directly to me. The pizza vouchers would be especially welcome. :-)"

Building Samba

Obviously, you must first get a copy of the source code. The version we found was a compressed `tar` file named *samba-1.9.15p8.gz*. You can find this source archive at the Samba home page *(http://lake.canberra.edu.au/pub/samba/)* or from our FTP site *(ftp.cobb.ziff.com/pub/cobb/solaris/journals/mar97)*.

NEWS FLASH: At the time of this writing, the latest version is 1.9.16p9. We've placed this version on our FTP site along with 1.9.15p8. The article was written with 1.9.15p8, but the same steps should work with 1.9.16p9.

Normally, when we download applications on the network, we place them in our */apps/Archive* directory. From there, we extracted it into the */apps/samba-1.9.15p8* directory with the following commands:

```
# cd /apps
# gunzip Archive/samba-1.9.15p8
# tar xf Archive/samba-1.9.15p8
```

Since Samba is freely available for many platforms, you must configure *Makefile* for Solaris. To do so, you need to make two or three changes.

First, you must specify the base directory in which you want to place the resulting executable files, libraries, and `man` pages. The default directory is */usr/local/samba*. If you want to change this, edit the file *source/Makefile*, locate the definition of the `BASEDIR` variable, and change it to the directory in which you want to place the installed version of Samba.

Next, you must configure the `FLAGSM` and `LIBSM` definitions for SunOS 5 (the base operating system for Solaris 2.x). To do so, search for `SUNOS5` until you find the following four lines. Delete the # symbol from the start of the blue lines.

```
# This is for SUNOS5 (also known as Solaris 2)
# contributed by Andrew.Tridgell@anu.edu.au
```

```
# FLAGSM = -DSUNOS5 -DSHADOW_PWD-DNETGROUP
# LIBSM = -lsocket -lnsl
```

Finally, you must select the compiler you're going to use. If you're using Sun's compiler, you need not do anything special. If you're going to use GNU `gcc`, you'll need to find the following lines and remove the # symbol from the start of the line in blue.

```
# You will need to use a ANSI C compiler. This
➥means under SunOS 4 you can't
# use cc, instead you will have to use gcc.
# CC = gcc
```

Now you're ready to create the executable. To do so using `gcc`, just type `gmake`. When you do so, `gmake` will compile and build the Samba programs and libraries. As it does so, it will print the name of each file it's compiling and linking.

Installing Samba

Compiling Samba doesn't install the programs; it just creates the binary images in the source directory. To actually install Samba, you'll execute the command:

```
# gmake install
```

This puts the executables in the */usr/local/samba/bin* directory, or the directory you specified in the `BASEDIR` variable if you changed it in *Makefile*.

Configuring Samba

Configuring Samba for an exotic setup can be challenging, as over 100 configuration options exist. Luckily, for standard configurations, you must understand only a few of them. Here, we're going to show you how to publish three resources. The Samba documentation and associated files give examples for other configurations as well.

You configure Samba by editing the file *lib/smb.conf* located in the base Samba directory. This is a text file, and any line starting with a semicolon is a comment.

To describe a resource, you create a section that specifies the resource. The section header is a line that starts with an open bracket followed by the resource name and a close bracket. The rest of the section is a list of key value pairs. The following four lines describe a typical global section. The global section is a special section that describes some parameters not associated with any particular resource or values that will be used as defaults in a resource definition.

```
; Set the default values
[global]
    workgroup = FPU
    guest account = Test
```

Here, we have a comment, the section header `[global]`, and two key value pairs. The first one, `workgroup = FPU`, tells Samba that the computer resides in the FPU workgroup. (The SMB protocol allows you to logically segment a network into workgroups.) The next one, `guest account = Test`, tells Samba that any service that uses guest access will use the Test account on the Solaris machine for the access privileges and password.

Setting up a publicly shared directory

The first service we're going to set up is a shared directory that will give read/write access to anyone on the network. To do this, we create a section for a new resource, which we'll call *share*, as shown here:

```
; Public file area
[share]
    comment = Public file area
    path = /export/home/share
    guest ok = yes
    writable = yes
```

As you can see, we have four key value pairs. The first specifies some text that Windows users will see when they execute the `net view` command, so use this to describe the resource. The next specifies the directory on your machine that you're going to share with all the SMB client machines on your network. Next, we tell Samba to use the guest account to access the resource. Finally, we tell Samba that we're allowing write access to the directory.

Using this definition, when users try to access the *//Ringo/share* resource, where *Ringo* is the host name of the computer, they'll have to specify the password to the Test account to gain access. Once they do so, they have complete access to the directory, assuming that the Test account does.

Adding access to home directories

If your users use the same login names on Windows as they do in Solaris, then you can

provide access to the users' home directories to the Windows machines. To do so, use the following section definition:

```
; Home directory access
[homes]
    guest ok = no
    read only = no
```

Samba treats the section named homes specially. It automatically maps the path to the home directory for the specified user. Here, we specify that the guest account doesn't have access and that the user may read or write to the store (i.e., isn't read only).

This is all you do if your users have the same login name on both Windows computers and your Solaris boxes. If they don't, you must read the Samba documentation on name mangling and let Samba translate the names.

Allowing access to a printer

Another operation that you'll probably want is to allow the Windows computers access to your printers. To do so, you must perform some extra steps. First, you add some commands to the global section to set the default values for printing on your system. Figure A shows our new global section with printer support.

Figure A

```
[global]
  guest account = Test
  workgroup = FPU
  print command = lp -c -d%p -o nobanner -o nofilebreak %s; rm %s
  printing = sysv
  load printers = yes
  printcap name = /usr/local/samba/lib/printers
```

We added the lines in blue to support printer sharing.

The first added entry specifies the command to execute to print the file. Here, we use the standard `lp` command to print the file. When your Windows application prints to Samba, Samba creates a temporary file to hold the data, then prints it. Since Samba doesn't delete the temporary file, *you* must, which is why we add the `rm` command after the `lp` command. Samba replaces the `%s` with the name of the temporary file, so the `lp` and `rm` commands know which file to print and remove, respectively. Similarly, Samba replaces `%p` with the printer name, so the `-d%p` option tells `lp` which printer to use.

The `print` command should execute as quickly as possible, so as not to hold up other Samba requests. But the printer won't finish printing the file before Samba executes the `rm` command, so we use the `-c` option to tell the `lp` command to make a copy of the file and print that copy. This way, when we remove the file with the `rm` command, `lp` can still print. The other options simply set the preferences we want for the printer.

The next entry tells Samba that we're using UNIX System V style printing, which is the default under Solaris. The third additional entry tells Samba that all the printers in the *printcap* file are available for browsing.

The final entry tells Samba the name of the file that contains all the printers to make available to the network. On many other UNIX systems, a file named */etc/printcap* contains an entry for each printer on the system. Solaris doesn't use a file with this format, so we told it to use the file */usr/local/samba/lib/printers*. We created this file using the following command (which we found in the *examples/simple/smb.conf* file in Samba's root directory):

```
$ lpc status | grep ":" | sed s/:/\|/
➥>/usr/local/samba/lib/printcap
```

Now that the global section is taken care of, we must create a printers section to tell Samba what to do with the printers. Figure B shows the printers section from our *smb.conf* file.

Figure B

```
[printers]
  comment = All Printers
  browseable = no
  printable = yes
  guest ok = yes
  writable = no
  create mode = 0700
```

This configuration section tells Samba the privileges of the printers and who may access them.

The first entry gives Samba, and the rest of the world, a description of the service. The next entry sets the browseable flag to no, so no other user may examine the outstanding print jobs. The next entry tells Samba that you can put files to print in the specified directory. You *must* set the printable flag in the printers section in order to print! We tell Samba to use guest account security, since a printer is a low-risk service. Next, we set the writeable flag to no, so no one writes an arbitrary file in our */tmp* directory. Finally, we set the create mode to 0700, so that the owner can do anything with the print file, but no one else can.

Testing our configuration

Once you create your configuration file, you should execute Samba's `testparm` command. If your *smb.conf* file contains any errors, `testparm` will alert you to them. If you don't have any errors, then you're ready to start Samba.

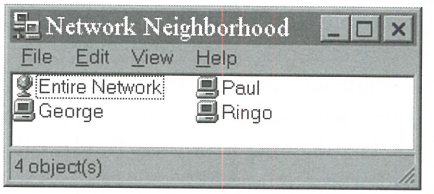
Starting Samba

To test Samba, you can start it by executing the `smbd` and `nmbd` programs with the `-D` switch. If executing `testparm` didn't show any errors, then go ahead and start Samba by typing

```
/usr/local/samba/bin/smbd -D
/usr/local/samba/bin/nmbd -D
```

Once you do, Samba should be running. Now test it by going to a Windows 95 computer and double-clicking the Network Neighborhood icon. When you do, you'll see all the computers in your workgroup, as shown in Figure C. (Please note that your Windows computer must use the TCP/IP protocol to access our Samba servers.)

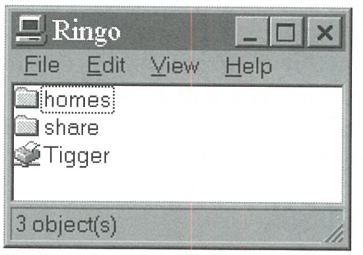
Figure C



George and Paul are Windows 95 machines, and Ringo is our Samba server.

Success! Now Ringo, our Solaris 2.5 machine, appears as just another computer on the Windows network. If we double-click the Ringo icon, we'll see the services that Ringo exported, as shown in Figure D.

Figure D



Our smb.conf *file exports these resources from Ringo.*

Doesn't that look wonderful? You can access the share directory simply by double-clicking on the share icon. When you do so, you must specify the password for the Test account. (In Windows 95, you can use a check box to tell Windows to automatically supply this password for future login sessions, so you won't have to do so again.)

Making Samba start automatically

Now you have one last step: Since Samba works properly, you want it to start each time you start your Solaris computer. You have two easy ways to do so. First, you can create a script that starts Samba and place this script in your start-up script area. Or (and we prefer this method) you can edit your networking files so that Samba starts whenever another computer tries to access files from your computer.

This method requires that you change two files. First, add the following two lines to your */etc/services* file:

```
netbios-ssn      139/tcp
netbios-ns       137/udp
```

Then, add these two lines to your */etc/inetd.conf* file:

```
netbios-ssn stream tcp nowait root
➥/usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root
➥/usr/local/samba/bin/nmbd nmbd
```

Now, each time you start Solaris, it's ready to be a happy member of your Windows network community.

Alternative products

Sun has a similar product to Samba, with more features, called Solstice LAN Client for Solaris. While you can access files on Windows networks with Samba, it's through an FTP-like interface. The Sun product allows you to directly access the files on the Windows network. For more information on this product, see the Web page at *http://www.sun.com/solstice/Networking-products/LANClient.html*.

If you'd rather run software on the Windows side, you can get Sun's PC-NFSpro v2.0 product. This product allows you to use the NFS protocol on your Windows machines, and you can then NFS-mount file systems you share on your Solaris computer. For more information on this product, see the Web page at *http://www.sun.com/solstice/Networking-products/PCNFSprov2.0.html*.

Conclusion

Samba is a very complex tool and offers you more flexibility than you'll probably require. If you must share files and printers with a network of Windows computers, it's a great tool.

Samba allows your Solaris computers access to files and printers on your Windows network as well, via the `smbclient` command. If you want to do this, or any other special operation, you'll have to dig through the documentation for examples. Luckily, there's a large community of people who use Samba and several news groups and mailing lists that you can read for configuration information. ❖

Partitioning a disk with Solaris x86

By Marco C. Mason

You keep adding users and software to a system and what happens? Yes, that's right, you run out of disk space. Now you must add a disk drive to your Solaris x86 box. To do so, you'll perform all the steps described in the article "Adding a Disk Drive to Your Computer," on page 11. However, on a Solaris x86 system, you must first perform an additional step: You need to partition your disk drive.

Another partitioning step?

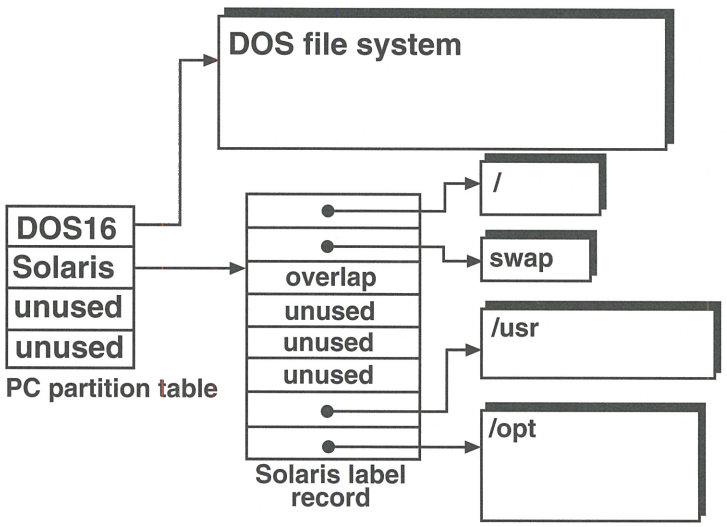
If you've read that article, you may be wondering why we're talking about partitioning the disk drive in this article. Didn't we cover partitioning in that article? It turns out that you must partition the drive twice because UNIX and PC-compatible computers have different partitioning strategies. On a Solaris computer, you partition a drive into slices, on which you store file systems. A Solaris computer may use eight or more slices on a single disk drive.

A PC-compatible computer, however, uses a partition table that can support only four partitions on a single drive. (Actually, there's a special case that allows you to use more than four partitions, but we won't go into that here.) You use these partitions to separate the physical drive into one or more logical drives, often for different operating systems.

Most operating systems that run on a PC-compatible don't recognize Solaris' disk-partitioning scheme, and Sun didn't want to modify Solaris to use the (rather constrained) partitioning scheme traditionally used on PCs. So a Solaris x86 computer uses the traditional x86 partitioning scheme to create a Solaris partition, then Solaris further subdivides that partition. Thus, you get two partition tables when you run Solaris x86: The BIOS partition table and Solaris' own partition table, which partitions only the Solaris partition.

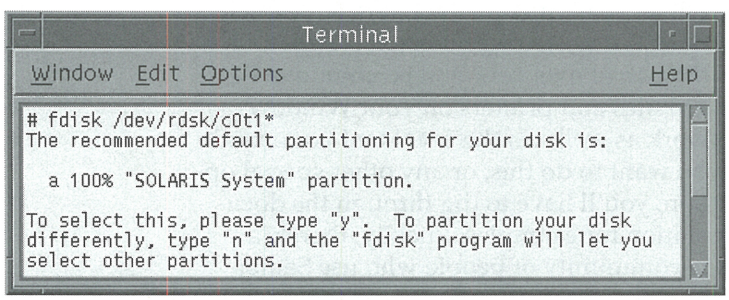
To illustrate the dual partition tables in use with Solaris x86, Figure A shows a drive partitioned to hold a DOS partition and a Solaris partition. The Solaris partition is sliced into /, swap, */usr*, and */opt* file systems. The box on the left shows the PC-compatible partition table that we're manipulating in this article. The box to its immediate right is the Solaris partition table, which we manipulate in the previously mentioned article.

Figure A



This hard drive is partitioned for DOS and Solaris.

Figure B



If you're running `fdisk` on a drive with no partitions, it asks you if you want to dedicate the entire disk to Solaris.

Running fdisk

You use the `fdisk` command in Solaris to partition a disk drive. To partition a disk, you start `fdisk` and tell it which raw disk drive to partition. If you look in the */dev/rdsk* directory, you'll notice that there are no entries for disk drives, just partitions and slices. All you do is invoke the `fdisk` command with the first partition (i.e., partition 0) of the drive you want to partition. Thus, to partition SCSI drive #1, you enter the command

```
# fdisk /dev/rdsk/c0t1d0p0
```

A single Solaris partition?

If you're running `fdisk` on a disk drive that contains no partitions, `fdisk` asks if you'd like

to dedicate the entire disk to Solaris, as shown in Figure B. If you enter *y*, then `fdisk` partitions the disk to a single Solaris partition and exits.

Supporting multiple OSes?

If the disk drive already contains partitions, or you select no in the previous query, you'll see the `fdisk` screen shown in Figure C. Here, you can see that we haven't created any partitions yet. At this point, you can create and delete partitions as needed.

Creating partitions

When you select 1 to create a partition, `fdisk` prompts you for the type of partition you want to create. Table A lists the partition types that `fdisk` knows how to create.

Table A

Choice	Partition Type
1	Solaris
2	UNIX
3	PCIXOS
4	Other
5	DOS12
6	DOS16
7	DOSEXT
8	DOSBIG

The `fdisk` command can create multiple partition types.

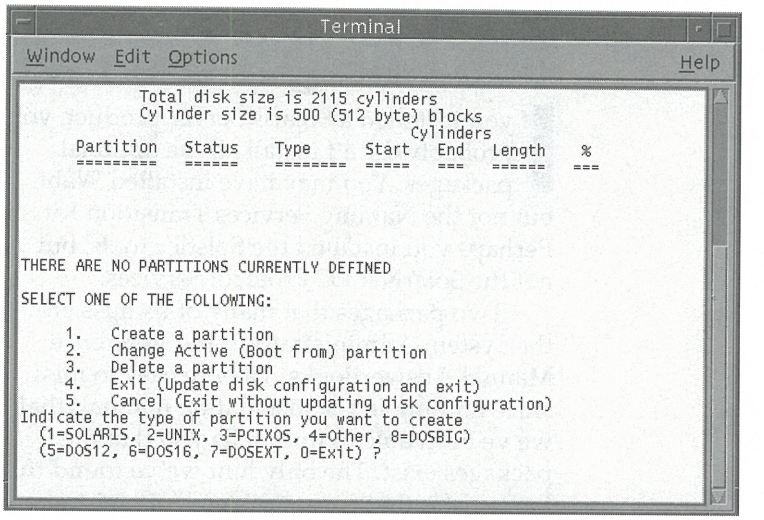
As you can see, Solaris knows how to create one partition for UNIX in general, one for PCIXOS, and four types of DOS partitions. We suggest that you use Solaris' version of `fdisk` to create only the Solaris partition for your system and use the disk-partitioning tool of other operating systems to create the other partitions. However, if you're familiar with disk partitioning in multiple operating systems, you can use the Solaris `fdisk` command to do most, if not all, of the work.

Please note that you may have only one Solaris partition on a drive. Solaris won't allow you slice up a second Solaris partition on the same drive. However, you can mount the second partition as a single file system if you desire. For more details on this procedure, see the article "Subdividing Your Hard Disk for Solaris x86" in the June 1996 issue.

Deleting partitions

If you're reusing or reconfiguring a disk drive, you may need to delete one or more partitions.

Figure C



The main screen allows you to create and delete partitions.

Since you can't resize a partition, if you must make a partition larger, you may need to delete two partitions, then re-create them using different sizes. Please note that PC-compatibles support only four disk partitions.

(Remember that exception that we mentioned earlier in the article? Well DOS can use multiple partition tables just as Solaris x86 does. The DOSEXT partition type tells DOS to add a partition table for splitting the DOSEXT partition into multiple pieces. This is how you can partition a hard disk with DOS to have more than four drive letters.)

Setting the active partition

When you boot your PC, it goes to the first hard drive and loads the operating system found on the active partition. Normally, you won't have to worry about this, since Solaris is already running. We're just adding additional disk drives, on which the PC will ignore the active partitions.

Conclusion

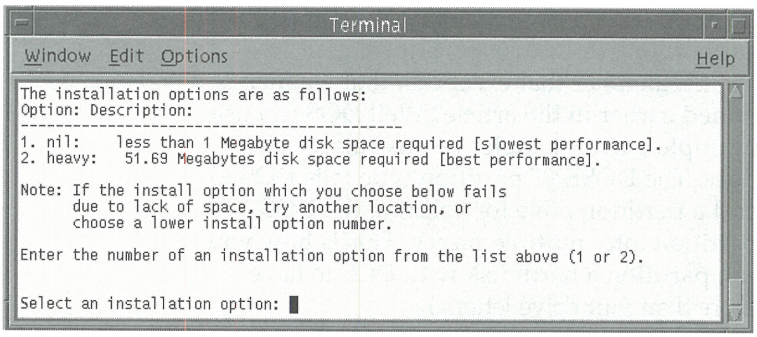
On a PC-compatible computer, Solaris uses two partition tables. The first allows you to cut the disk drive into as many as four pieces. You can use one of these pieces for Solaris and the others for other operating systems, such as DOS or OS/2. Once you've created a Solaris partition on your disk drive, Solaris may then treat the Solaris partition as a physical disk drive and allow you to slice it into multiple file systems as we described in the accompanying article "Adding a Disk Drive to Your Computer." ❖

Don't forget the rest of the AnswerBooks

If you installed a Solaris Server product, you probably didn't install all the optional packages. You may have installed Wabi, but not the Naming Services Transition Kit. Perhaps you installed the Solstice tools, but not the SolarNet PC Protocol Services.

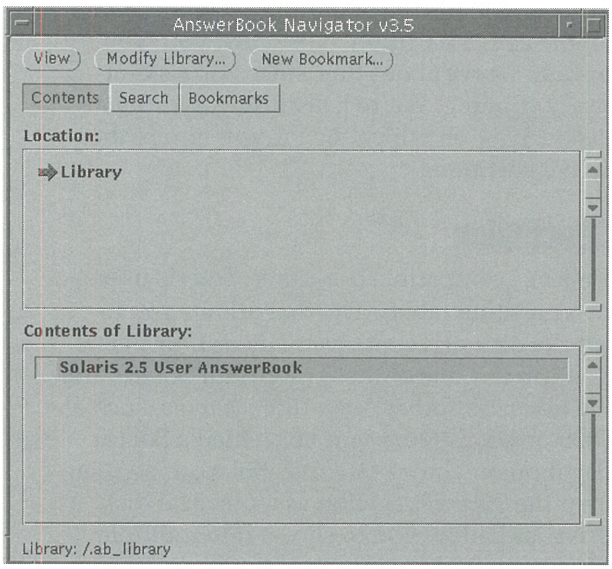
Two packages that many of us miss are the System Administrator's and Reference Manual AnswerBooks. They're easy to miss, since nothing in the installation manual (that we've been able to find) tells you that the packages exist. The only hint we've found that they exist is the phrase "System Administrator's AnswerBook" on the Solaris Server CD-ROM. In this article, we're going to show you how to install the System Administrator's and Reference Manual AnswerBooks for Solaris 2.5.

Figure A



Use the pkgadd command to install the AnswerBook packages.

Figure B



If you don't enable the other AnswerBooks, you'll have access only to the Solaris 2.5 User AnswerBook.

Disk space requirements

The System Administrator's AnswerBook takes about 52MB, and the Reference Manual AnswerBook takes about 93MB. Because these packages are so large, Sun allows you to install them in one of two ways. The first way, nil, allows you to keep the information on the CD-ROM. This takes very little disk space, but you must have the CD-ROM containing the AnswerBook data mounted to use the AnswerBooks. The second mode, heavy, puts all the information on your disk drive. This makes the AnswerBook respond better, and it doesn't tie up your CD-ROM.

For this article, we're going to install the AnswerBook package in heavy mode. This way, you'll get the speed of a hard disk. If you don't mind mounting the CD-ROM each time you use the AnswerBooks, you may want to select the nil mode.

To install the AnswerBook packages, mount the appropriate CD-ROM. Next, you must find the appropriate packages. Solaris Server users should mount the Solaris 2.5 Server Supplement 1.0 CD-ROM; the Answer Books are in the /cdrom/solaris_2_5_ server _1_0/SysAdmAB2.5 directory, as shown here:

```
# cd /cdrom/solaris_2_5_server_1_0
# ls
Copyright  NSKit-1.2  PPS1.1    SysAdmAB2.5
# cd SysAdmAB2.5
# ls
Copyright common
```

The *SysAdmAB2.5/common* directory contains the AnswerBook packages:

```
# cd common
# ls
SUNWaadm SUNWaman
```

To install these packages, enter the command:

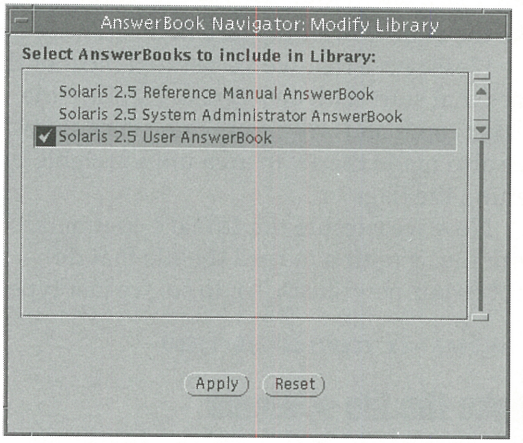
```
# pkgadd -d . SUNWaadm SUNWaman
```

When you do, **pkgadd** will install *SUNWaadm* first, followed by *SUNWaman*. To install these packages, **pkgadd** will ask you three questions. First, it will ask whether you want to use the nil or heavy mode, as shown in Figure A. Choose option 2 to start a heavy installation.

Next, **pkgadd** wants to know the parent directory for the AnswerBook directory. When you install Solaris, it will automatically put the

AnswerBook code in */opt*, so enter */opt* as the directory name unless you explicitly placed the AnswerBook software elsewhere.

Figure C



From this screen, select all the AnswerBooks you want to access.

Finally, `pkgadd` will tell you that it will execute a script with superuser privileges to complete the installation. Reply with *y* to tell it to go ahead, and `pkgadd` will install the first package. Once `pkgadd` finishes installing *SUNWaadm*, it will ask the same questions for installing *SUNWaman*.

Using the AnswerBooks

Once you install the AnswerBook packages, you still must enable them. Until you do so, you'll have access only to the Solaris 2.5 User AnswerBook, which was installed when you installed Solaris.

To enable the AnswerBooks, start the AnswerBook and press the Modify Library... button, shown in Figure B. When you do so, you'll see a list of all the AnswerBooks installed on the system, as shown in Figure C. Simply select each AnswerBook to which you want access (so that a check mark appears next to them) and press the Apply button. ❖

SYSTEM ADMINISTRATION

Adding a disk drive to your computer

By Marco C. Mason

Sooner or later, you'll have to add a disk drive to your computer. The one nice thing is that as time goes by, disk drives get cheaper. Four years ago, a 340MB SCSI drive could cost over $800. Now, it's getting hard to find drives smaller than 1GB, and even these cost $250 to $400, depending on their speed.

In this article, we'll show you a simple procedure for preparing your computer to use a hard drive you just added to your computer. Because of the many differences in computer systems, we're not going to cover the vagaries of hardware installation.

Please note that if you're running Solaris x86, you must follow the instructions provided in the accompanying article "Partitioning a Disk with Solaris x86," on page 8, before starting this procedure.

Design your file system layout

After you install the hard disk drive, you must decide how to partition it. Do you want to add any swap space? Which file systems will you place on the drive, and how large will they be? Once you answer these questions, you're ready to go to the next step, creating the slice table.

For the purposes of this article, we'll create a simple system. We're adding a 515MB drive to our system. We'll use 30MB to extend our swap area, and the rest will be for our */usr/local* file system that was overflowing our */usr* file system.

Create the slice table

Now that you've designed your file system layout on the disk, you must partition the disk. One of the easiest ways to do so is to execute the `prtvtoc` program to print out the disk parameters and the default slice table for the drive, edit the file, then use the `fmthard` command to read the file and partition the drive.

Since we want to edit the results of the `prtvtoc` command, we'll pipe the results of its output to the file *x*. For the purposes of our example, we installed the disk drive at SCSI ID #1, so we use the following command:

```
# prtvtoc /dev/rdsk/c0t1d0p0 >x
```

In case you're not familiar with naming disk drives, partitions, and slices in Solaris, see the sidebar "Disk Slice Names in Solaris" on page 13.

With the drive we used in our example, `prtvtoc` returned the results shown in Figure A. Please note that since we're reusing a Solaris drive, some slices are already defined. We first edited the file to remove all slice definitions *except the one defined for slice 2*. You can't delete the definition for slice 2 because that slice tells Solaris how big your disk is. Never, ever change the information in slice 2 on a disk drive.

Solaris wants each partition to start on a cylinder boundary, so we must select slice sizes such that the next slice will begin on the appropriate boundary. As you can see in Figure A, each track holds 125 sectors, and each cylinder has four tracks. Thus, all cylinder boundaries should occur at multiples of 500 (125*4) sectors. Since the size of a sector is 512 bytes, a cylinder holds 512*500, or 256,000, bytes. Therefore, four cylinders is about a megabyte.

Since we want a 30MB slice for swap, we want the first partition to be 30*4, or 120, cylinders (60,000 sectors). We can then use the remainder of the disk for the */usr/local* file system. Once we're finished, our file should look like the one shown in Figure B.

Please note that we've omitted most of the lines that start with an asterisk (*), since the `format` command will ignore these lines anyway. We also made the swap area unmountable by setting the flags to 1.

Now we execute the `fmthard` command with the `-s` option to read the file that describes our new slices. To do so, we just type

```
# fmthard -sx /dev/rdsk/c0t1d0p0
```

Make the file systems

Now that you have the drive partitioned the way you want, you must put a valid file system on the slices that you intend to mount. (You don't mount swap space, so you can bypass this step for partitions you're going to add to your swap area.)

The easiest way to make the file system is to use the `newfs` command. To do so, you simply specify the slice on which you want to put a file system and let `newfs` do the work. For our example, we only need to create a file system on slice 1, so we type

```
# newfs /dev/rdsk/c0t1d0s1
```

Creating the mount points

Of course, you must have locations to mount your new file systems. As you probably know, a mount point is simply a directory. For totally new file systems, you can simply create the directory. If you're adding a new disk drive to alleviate some space problems, as we are in our example, you must use a more complex procedure:

- Mount the slice at a temporary mount point.
- Copy some directories to the new file system.
- Delete the directories you just moved.
- Unmount the file system from the temporary mount point.
- Mount the slice at the new mount point.

Figure A

```
* /dev/rdsk/c0t1d0p0 partition map
*
* Dimensions:
*     512 bytes/sector
*     125 sectors/track
*       4 tracks/cylinder
*     500 sectors/cylinder
*    2114 cylinders
*    2112 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector    Count     Sector  Mount Directory
       0      4    10          0    1056000   1055999   /usr
       2      5    01          0    1056000   1055999
```

You can use the `prtvtoc` command to create a file suitable for use by the `fmthard` command.

Figure B

```
*                          First   Sector    Last
* Partition  Tag  Flags    Sector  Count     Sector    Mount Directory
       0      0    01         0     56000   55999
       1      0    00     56000   1000000   1055999   /usr/local
       2      5    01         0   1056000   1055999
```

Now we've modified our data file to describe our desired partitions.

Figure C

```
#device          device              mount       FS    fsck  mount    mount
#to mount        to fsck             point       type  pass  at boot  options
/dev/dsk/c0t1d0s1  /dev/rdsk/c0t1d0s1  /usr/local  ufs   1     yes      -
/dev/dsk/c0t1d0s0  -                  -           swap  -     no       -
```

Adding the blue lines to the /etc/vfstab *file tells Solaris to add our new partitions during boot.*

With the default installation, Solaris provides a temporary mount point named */mnt*, so you need not create one. So we'll mount slice 1 at */mnt* and copy all the files in */usr/local* to */mnt*.

```
# mount /dev/dsk/c0t1d0s1 /mnt
# cd /usr/local
# tar cf -; (cd /mnt; tar xf -)
```

You'll notice that we used the `tar` command to perform the copy. We did so to preserve all the ownership information about the files. Now, verify that all the files copied correctly. Once you've done so, delete the files at */usr/local* to free up your disk space. Then you can unmount the slice from */mnt* and mount it at */usr/local*.

```
# rm -r /usr/local
# umount /mnt
# mount /dev/dsk/c0t1d0s1 /usr/local
```

Adding the file systems to /etc/vfstab

You've finished all the hard work. Go ahead and test everything, so you can be sure that nothing is lost. Now all that's left is to add the appropriate entries to */etc/vfstab* so the file systems (and swap area, if you created one) will automatically mount whenever you restart Solaris. For our example, we had to add the two blue lines shown in Figure C to */etc/vfstab*.

Conclusion

Adding a hard drive to your system needn't be a problem. Just follow this cookbook procedure, and you should be able to set up your hard drive soon after you get the hardware installed successfully. (While testing this article, we did the procedure often enough that we can now do it in just a few minutes, less the time needed for copying file systems.) ❖

Disk slice names in Solaris

You may not be aware of the conventions that Solaris uses to name disk drive slices and partitions. We're going to give you a brief overview, so you'll be familiar with it.

The driver names of the disk slices and partitions in Solaris are located in the */dev/dsk* and */dev/rdsk* directories. You use the */dev/dsk* directories when you want to access a specific file system, and you use the */dev/rdsk* names when you want to access a "raw" device.

Both the */dev/dsk* and */dev/rdsk* directories contain files named like one of these four cases:

	Slice	Partition
with target	*cCtTdDsS*	*cCtTdDpP*
without target	*cCdDsS*	*cCdDpP*

The first line shows how to access a slice or partition on a controller that uses target addresses, like SCSI. The value *C* specifies the controller number, *T* specifies the target address on the controller, *D* specifies the drive number, *S* specifies the slice number, and *P* specifies the partition number. The second line shows drive and partition numbering for controllers that don't support target ID numbers, like IDE controllers.

Please note that the names for partitions exist only on Solaris x86 machines. These partitions refer to the BIOS partitions for IBM-PC compatible machines. There's no equivalent on a SparcStation, so you won't see these entries on SparcStations.

QUICK TIP

How much space will a package take on your hard disk?

It's happened to all of us. You're trying to install a package and you run out of space on the file system you're installing to. Wouldn't it be nice if you could find out how much space a package was going to take *before* you installed it? Usually, vendors tell you how much space the packages will require—but in many cases, they don't.

Getting an estimate

If the vendor doesn't tell you how much space a package is going to take, you'll either have to install it and see what happens or try to get an estimate. It turns out that if the package is on a disk drive, you can very easily estimate the amount of space required by the package. To do so, simply go to the

directory that holds the package and execute the command

```
# du -s pkgname
```

where *pkgname* is the name of the package you want to check.

As an example, let's look at the directory */cdrom/wintertime.x86/Perl*, which contains several packages that comprise a Perl distribution:

```
# cd /cdrom/wintertime.x86/Perl
# ls
Perldocx Perlext Perlmisc Perlperl Perlscrip
Perlsrc
```

If we want an estimate of how much space the source code would take on our hard drive, we can execute the command

```
# du -s Perlsrc
7287   Perlsrc
```

This shows that the Perlsrc package will take approximately 7,287 blocks on our file system.

The number isn't exact

The value returned by the `du` command should be considered an estimate, rather than an exact package size. A package may take less space than indicated by `du` for several reasons. First, some of the files may not be installed in the final distribution: Perhaps an installation script will delete some of the files on installation. Other files may be compressed to save disk space and expanded after installation.

Another reason that a package may not take as much space as `du` indicates is that it may put some files in a directory where identical files already exist. GNU tools, for instance, often have some overlap, so if you already have some tools loaded, other tools may take less space because of the redundancy.

Conclusion

This isn't a foolproof method for finding out how much space a package is going to consume, but at least you can get a good estimate of the size of the package if the vendor neglects to tell you. ❖

LETTERS

Operating on a file in a pipeline

I'm a longtime Sun employee, presently a staff engineer in SunSoft Software Quality Process Engineering. I'd like to notify you of a nasty error in the January 1997 issue of *Inside Solaris*. On page 9, in blue, is the example

```
sort < /LargeFile | uniq > /LargeFile
```

Perhaps the author intended the output to go to a different file, such as */LargeFile.sorted*, but as written, the shell will merely empty */LargeFile*! Then `sort` and `uniq` will find empty input files.

A correct example would be

```
sort -u -o /LargeFile /LargeFile
```

The `sort` command knows how to read an input file without needing the shell to open it via <, write a file without needing the shell to open it via >, and avoid outputting anything before all input is read; `sort` also knows how to limit output to unique lines (`-u` option).

W. Dean Stanton
Mountain View, California
via the Internet

You're perfectly correct. The original example with the article didn't have the bug—it was inserted when I mistakenly thought I'd clarify it. Ouch!

The example fails as you cited because when the shell starts processing the command line, it breaks the command

```
sort < /LargeFile | uniq > /LargeFile
```

into two parts at the vertical bar, effectively turning it into two commands like this:

```
sort < /LargeFile > Pipe
uniq > /LargeFile < Pipe
```

where *Pipe* is an intermediate pipeline. Then the shell starts two processes. For the first process, it opens */LargeFile* for the input stream and *Pipe* for the output stream, and starts executing the `sort` command. For the second process, it opens *Pipe* for the input stream and */LargeFile* for output, and starts executing `uniq`.

However, once it opens */LargeFile* for output, it erases all the data that was there. Thus, the `sort` and `uniq` commands encounter an empty stream of data, and the result is a rather inefficient way of erasing */LargeFile*. ❖

DON'T PANIC!

What about the year 2000?

As you've probably seen, the popular computer press is abuzz with a new impending catastrophe: the year 2000. If you pay attention to all the doomsayers, life on earth shall cease as people lose their life insurance, bank accounts, and frequent flier miles.

Solaris is safe

The problem is real and it's severe, but it won't impact Solaris 2.x itself. At least not at the year 2000. Solaris stores the date by keeping track of the number of seconds since January 1, 1970 GMT. Since Solaris uses a 32-bit integer, we won't have any problems until 2038.

Long before then, the 64-bit version of Solaris will be released. Assuming that they use the same technique, but with a 64-bit integer, the date won't roll over until well after the earth falls into the sun.

Your applications, however...

While Solaris is safe from the problem, your applications may not be so safe. If they use the same UNIX date format, there's a good chance you can sleep soundly. The usual culprits fingered as the problem for the year 2000 are programs that store their dates by specifying only the last two digits of the year. Many of these programs were written in the days of punch cards or small disk drives, where every character counted. In those days, people didn't imagine that their programs would still be running in the year 2000.

Business programs, however, have a habit of growing and remaining in use. The longevity of some of those programs is astounding. The older the program is, the more probable that it uses a two-digit date. Unfortunately, the older the program is, the more expensive (on average) it will be to fix.

Conclusion

If you have any applications that must be updated, the time to do so is now, while you have the chance. There's only a limited amount of time to fix it. (At the time of this writing, there are only about 750 working days until January 1, 2000!) ❖

QUICK TIP

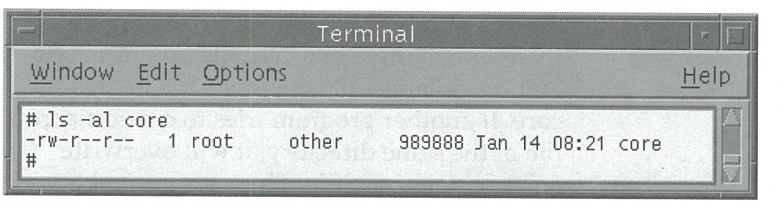
Who dumped core?

If you manage multiple systems for many users, especially software developers, you may be familiar with this problem: *core* files abound everywhere. As Figure A shows, these core files can take a substantial amount of disk space, especially when you have many of them.

As you probably know, Solaris generates a core file when a process crashes. These core files contain lots of interesting information to a software developer who's trying to fix the program. The core file shows what the program was doing when it terminated. Using a debugger, such as `adb`, you can examine *exactly* what the program was doing at the time of the crash.

As we mentioned in the article "Don't Clutter Your System With core Images!" (in the July 1996 issue), you can use the `ulimit -c 0` command to tell Solaris not to generate core files. (Each user who doesn't want to generate core files must execute this command each time he or she logs in to the computer.) However, if some program on your system is generating these core files, you might want to save them to help the developer repair the problem. For example, a vendor may have sold you a program that crashes on your sys-

Figure A

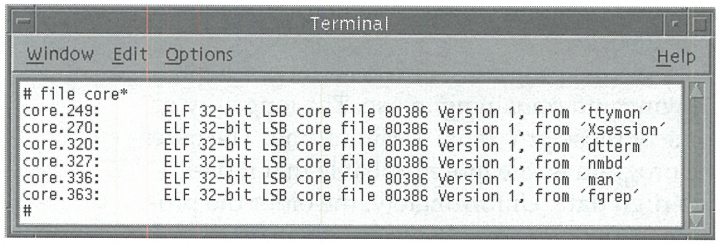


Each core file can waste a significant amount of space, especially if you have many of them.

PERIODICALS MAIL

Please include account number from label with any correspondence.

Figure B



You can use the `file core` command to find the names of the programs that generated a core file.

tem. If you can provide the vendor with some core files, you may help resolve the problem.

If this is the case, the first step is finding out just which program generated the core file. Usually, the command `file core` will give you that information. (As long as you're running a program in the ELF format, the `file` command can decipher the internal structure of the core file to find the name of the program that generated it.) Figure B shows an example where we manually generated a set of core files.

One thing we should mention about Figure B: All our files are named *core.xxx*, where *xxx* is a number. We did this to give more than one example of a core file. In normal operation, Solaris will generate a file named core. If another program tries to create a core file in the same directory, it will overwrite any previous core file. Thus, for normal operation, you'll only need to specify file *core* to find the name of the program, but in our figure we used `file core*` to get the information on all our test files.

Just in case you'd like to experiment with core files yourself, they're easy to generate: You can use the `gcore` *procID* command to generate a core image of a running program. In this command, *procID* specifies the process ID of a running program. To create our test files, we simply executed the `ps -ef` command to find some interesting looking processes and used `gcore` to generate the core images. When you do so, `gcore` will generate a file named *core.xxx*, where *xxx* is the process ID you specify.

If you're trying to collect information on core files, and you have many users on your system, you can find the specific files you want with a command like this:

```
# find / -name core | xargs file
```

This command will find all the files named *core* on your system (if you're logged in as the root user) and print the name of the program that generated the core file. You can copy the ones you're interested in to an archive, then delete them all with the command:

```
# find / -name core | xargs rm
```

If you have software developers on your system, you might not want to delete all core files because they may be in use by your developers. Instead, you can set up a time limit and delete core files only when they reach a certain age. You can do so by adding the `-mtime` option to the `find` command, like this:

```
# find / -name core -mtime 7 | xargs rm
```

This command deletes all the core files older than seven days. This way, your software developers have a few days to use their core files or rename them to something other than *core*.

Conclusion

Those core files on your system don't have to go out of control. Armed with the information in this article, you can find out which program created the core files on your system and send the important ones to the software developers. Then, you can simply delete the rest of them. ❖