COBB

August 1997 • $11.50
VOL. 3 NO. 8

INSIDE SOLARIS™

Tips & techniques for users of SunSoft Solaris

in this issue

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

An introduction to the automounter system, part 1

When you're administering a tiny network, it begins simply: You can accommodate changes to your network by hand-editing some configuration files. So when you add a new host to your network, you edit the */etc/hosts* file of each computer that must access the new host. Similarly, if you want to share a file system among several computers, you edit the */etc/dfs/dfstab* file on the server to publish the file system, and you edit the */etc/vfstab* file on each client that wants access to the file system.

As networks continue to grow, they become harder to administer. As your network grows, it becomes more difficult to edit all the configuration files on all your hosts each time you modify the network. Making things even more difficult is the fact that larger networks are usually reconfigured more often than small networks. So as your network expands, the effort required to change the network increases, and the number of changes also increases.

For this reason, you must turn to some of the more complex network administration tools once your network grows large enough. For example, you might choose to use DNS to manage new hosts or NIS+ to manage just about everything.

In this article, we'll examine one tool that greatly simplifies the management of shared file systems—the automounter system. This system, composed of the `automount` program and the `automountd` daemon, allows you to easily share file systems between computers, allows users to move around the system, and minimizes the amount of administrative effort required to make changes to the network.

What are the benefits?

In order to illustrate the benefits of the automounter system, let's examine some scenarios that you'd encounter on a small network and see how the automounter system can help you simplify the situation.

User mobility

With the automounter system properly configured, your users can log in on multiple computers, with their home directory apparently traveling with them. By default, `automount` mounts your home directory in */home*. If you use this convention, then any computer on which you log in appears to be your real home directory, such as */home/marco*. It's easy to reconfigure automounter to place your home directories in another location if you use a different convention on your network.

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 ..... Jerry L.M. Phillips
Sudhir Wadhwa
Production Artists ..... Margueriete Winburn
Natalie Strange
Editor ..... Karen S. Shields
Publications Coordinator ..... Linda Recktenwald
Managing Author ..... Eddie Tolle
Editor-in-Chief of Online Publishing ..... Darren McGee
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

Of course, you could do this with NFS, but the administrative overhead is significant. Since you want the users to be able to log in on the new computer, you must ensure that the file system containing the user's home directory is shared from the appropriate computer. Next, you must ensure that this file system is mounted on every computer on which you want the user to have access. So you'll need to customize */etc/vfstab* on each computer you want users to be able to move between. With the automounter system, you still must share your file systems, but you can skip the hassles of customizing */etc/vfstab*.

Disk space management

These days, disk space is cheap, but still you don't want to waste it unnecessarily. If you have many identical SPARCstations, you also have multiple copies of many programs and data files. Figure A shows a segment of a network where each workstation has a combination of common data (programs, etc.) labeled A, and data and programs unique to the workstation, numbered 1, 2, and 3.

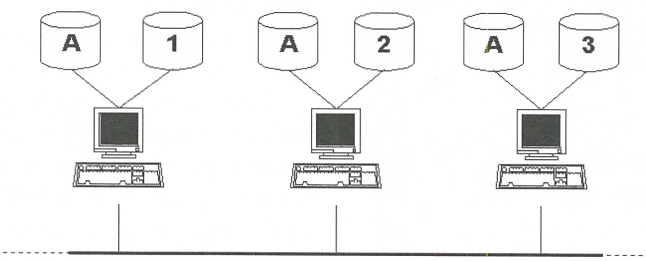
Think of all the wasted disk space! On my desktop, on a small installation, my computer uses 8MB to store just the */usr/bin* directory. Think of the space you could reclaim if you could consolidate all your `man` pages, */opt* directories, applications, and data!

Using NFS, you can consolidate by sharing the directories on a computer, using NFS to serve these directories, then making all the other computers mount these file systems on bootup, as shown in Figure B.

But what happens if your NFS server crashes? Everything grinds to a halt because all your programs and data are on the NFS server. Similarly, if a server is out of service, any client accessing that NFS server will hang on bootup while it tries to mount the file systems.

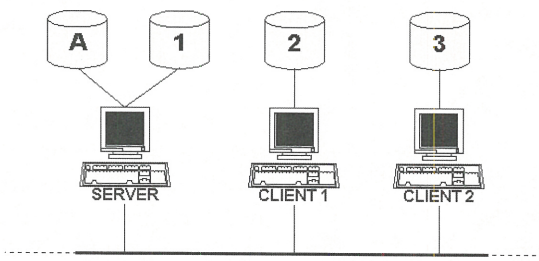
In small networks, many computers typically share data with many others. After a while, each computer seems to be NFS-mounting file systems from each other computer, causing a complex array of linked systems. In such cases, booting up all the computers on the network can take an excruciatingly long time as each computer

Figure A



We waste space on a network when common data is replicated on all computers.

Figure B



You can save a tremendous amount of disk space by sharing data and programs among computers.

tries to mount the file systems on hosts that aren't yet ready.

The automounter system alleviates these problems. First, it allows you to use multiple computers to host various file systems. If one server goes offline, then your system will simply get the data it wants from another server. You can use a few servers in different locations to share your applications and data, and all your workstations will have lots of extra disk space!

Another feature is that the automounter system mounts file systems only when they're accessed. So your workstations won't hang if (heaven forbid!) all your servers are offline after a power failure. However, any process accessing a file system that's offline will pause. But that won't stop the rest of the processes from doing their jobs.

The automounter system solves this problem by mounting file systems on demand, rather than on bootup. So until you access a file on a remote computer, your host doesn't mount it. This allows a network of computers to quickly boot.

How does it work?

We keep comparing the automounter system to NFS because NFS is the standard file-sharing technology for UNIX, and the automounter system is based on NFS. Basically, the system has two programs, `automount` and `automountd`. You can think of `automountd` as a really fast superuser who will watch certain mount points and continually `mount` and `umount` NFS file systems on demand. The `automount` command is our interface to `automountd`, which tells our fast superuser the system rules.

So if you execute the command

```
# cd /home/marco
```

the automounter system looks for a rule to tell it where to find the directory for marco. It doesn't matter if the directory is local or on another computer. If it's local, the superuser will create an invisible link to the local file system. If it's on a remote machine, the superuser mounts the directory in its own private area, then creates the invisible link.

So the automounter system is based on tried-and-true NFS technology. Since the daemon mounts file systems only when they're required, you needn't worry whether an NFS server is up or not when booting your computer. When you configure the automounter system, you can specify multiple locations for each file system, so the daemon can find one when you need it.

You tell `automountd` what to do and how to do it with the */etc/auto_master* file. Each time `automountd` starts (i.e., when you boot your computer), it reads */etc/auto_master* to find out which mount points it needs to manage and where the file systems are located.

The default */etc/auto_master* file, shown here, is probably good enough for most clients:

```
# Master map for automounter
#
+auto_master
/net    -hosts          -nosuid
/home   auto_home
/xfn    -xfn
```

The first two lines of */etc/auto_master* are comments, while the remaining four lines tell `automountd` where to get its operational rules. These sets of rules are called *maps*, because they map certain directory accesses to shared resources on your network.

The first line of interest in the */etc/auto_master* file is `+auto_master`. This line tells the daemon first to read the NIS or NIS+ auto_master map. If you're not using NIS or NIS+, don't worry; the line will be ignored. Otherwise, the automounter system will read any maps described using the NIS or NIS+ databases.

The next three lines specify a directory and the rules that tell `automountd` how to control it. When `automountd` starts, it takes control of the specified directory. Then, each time you try to access a file or directory in that file system, `automountd` examines the map to decide how to interpret your request.

Mapping rules

There are special maps, indirect maps, and direct maps. The line telling `automountd` to control the */net* directory is an example of one of the three special maps. The following line, specifying the */home* directory, is an example of an indirect map. The standard */etc/auto_master* file doesn't specify any direct maps.

Special maps

The entry that takes over the */net* directory uses a special map named `-hosts`. This special map tells `automountd` that each host on your network maps to a subdirectory in */net*. So if you try to go to the */net/Servalan* directory, `automountd` looks for a host named Servalan. If it finds Servalan, `automountd` asks for a list of

all the file systems that Servalan is willing to export to your machine. Then `automountd` mounts those file systems in the directory */net/Servalan*. On our computer, you'd see something like this:

```
# ls /net
# ls /net/Servalan
opt     export
# ls /net
```

As you can see here, the first time we asked for a directory listing of the */net* directory, the `ls` command didn't show anything. When we asked for a listing of */net/Servalan*, however, `automountd` stepped in and looked for a host named Servalan. Then it mounted the file systems Servalan was willing to share with us. When the `ls` command completed, it showed us the exported file systems. Our third `ls` command, showing the */net* directory, presents Servalan as a selection. If no one accesses anything in the */net/Servalan* file system for five minutes, then `automountd` will unmount Servalan's file systems.

The other two special maps are less interesting. The `-xfn` map allows you to access the Federated Naming System (FNS), and `-null` allows you to cancel a map for a directory.

Indirect maps

Now let's look at the line that starts with */home*. This line tells `automountd` that it's supposed to manage the */home* directory and that the rules it should use are in the file */etc/auto_home*. The format of the */etc/auto_home* file is similar to */etc/auto_master*. The first column specifies the name of the subdirectory of */home* that you're trying to access. If you want to specify any mount options for the file system, you specify them in the second column. The next column (two if there are no mount options, three otherwise) specifies the host and directory name where you want to map it.

Our */etc/auto_home* file, shown below, tells `automountd` that whenever someone accesses the */home/marco* directory, connect it to *Servalan:/usr/marco*. If someone requests */home/dean* instead, then `automountd` tries to connect to *ratbert:/usr/dmzigoris*. As you can see, the directory names in the rule needn't have any relationship to the subdirectory name. This can be handy when you try to integrate computers that have different user-naming strategies.

```
# /etc/auto_home

marco     Servalan:/usr/marco
dean      ratbert:/usr/dmzigoris
*         -nosuid punchy:/usr/&
```

The last line is the most interesting one. It illustrates two things. First, it demonstrates a cool feature of the automounter system: its ability to use wildcards. The `*` tells `automountd` to try the rule with any other subdirectory of */home*. The automounter replaces the `&` symbol with the current value of `*` to build the appropriate subdirectory of *punchy*. Thus, if you try to access user marti, `automountd` first checks to see if you specified marco. Since you didn't, it then checks to see if you specified dean. Again, you didn't. Then it checks to see if you specified anything. Since you did, it attempts to try the rule *punchy:/usr/&*. To do so, it first replaces the & with marti (the current value of *) and tries to attach to *punchy:/usr/marti*. We'll go into this in greater depth in our next issue.

The second interesting thing about this line is that we specified a mount option: `-nosuid`. This option is passed straight to mount by `automountd`, so it means the same thing it means when you mount an NFS partition. (In this case, it prevents a program on *punchy* from executing with the SETUID permission.)

Direct maps

A direct map is very similar to an indirect map, with one important difference: Rather than specifying a mapping of a subdirectory name to a directory on another host, it allows you to map any directory to another host. While this sounds powerful, it's less efficient than an indirect map because it makes `automountd` effectively manage more locations. The */etc/auto_master* file doesn't specify any direct maps by default. However, you can add a direct map by adding the following line to your */etc/auto_master* file:

```
/-        auto_direct
```

In an indirect map, you tell `automountd` to manage a directory for you, and you specify that directory in the */etc/auto_master* file. Since a direct map allows you to specify explicit directories, you needn't specify a directory for `automountd` to manage. The convention in this case is to use `/-` as a placeholder for the first column. Then you specify the name of your map file (located, by default, in the */etc* directory). In this case, we specified the file *auto_direct*.

Our */etc/auto_direct* file (shown next) contains only two entries: We first tell `automount` to use *punchy:/usr/man* whenever we refer to

the */usr/man* directory. Since the `man` pages are never written by anyone, we mount the file system read-only with the `-ro` switch.

```
# /etc/auto_direct

/usr/man-ro      punchy:/usr/man
/usr/local       -ro,-nosuid,-noguid
widget2:/usr/local
```

The second directory for which we're using a direct map is our */usr/local* directory, which we map to *widget2:/usr/local*. Since this directory contains software written in-house, as well as GNU and other software from the Internet, we also restrict the operation of the SETUID and SETGID programs.

Notes

There's much more to the automounter system than we can cover in a single article. In our next issue, we'll describe how the automounter system can use variable and wildcard substitution to help you select the appropriate network resources to connect to. You should definitely refer to the `man` page for the `automount` command for information on some of the more advanced features. You may also want to read the article "Improving Solaris 2.3 and 2.4 Performance With CacheFS" in our June 1995 issue to see how the automounter system interacts with the CacheFS system.

Conclusion

You can simplify some administration of your system by using the automounter system. If you need casual access to another host, you can often get what you need simply by linking to */net/hostname*, where *hostname* is the computer you want to connect to. With only a little editing of your */etc/auto_home* file, you can provide simple access to home directories on any computer on your network.

In the next issue, we'll continue our description of the automounter system. We'll illustrate some of the more advanced features available to you and give an example configuration that shows you how to take advantage of them. ❖

SYSTEM ADMINISTRATION

The basics of file-system tuning

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

UNIX system administrators, charged with the task of implementing space-hungry applications such as News, quickly learn the importance of maximizing all available disk space. In this article, we'll describe some simple techniques that will allow you to create a single, large partition and increase storage space on your disk drive(s). We're using Solaris 2.5 on a SPARC platform, but these techniques should work equally well on any Solaris 2.x computer.

Creating a single, large partition

A disk-intensive application, such as a news server, will quickly consume all the disk space you can throw at it. With the low cost of disk drives, it makes sense to put the news data on its own drive (or drives). Then you should dedicate the entire drive to a single file system to hold the application's data files. Using multiple slices can be wasteful and difficult to manage. It's wasteful because empty space or free inodes on one slice can't be used by another slice that's full.

To create a single, large partition, first use the `format` command to select a disk drive installed on your Sun SPARC. For our example, we're using a Sun 2.1GB disk drive, which just happened to have data on it from a previous installation.

```
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
0.       c0t0d0 <SUN2.1G cyl 2733 alt 2 hd 19 sec 80>
         /iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/
         ➥esp@5,8800000/sd0,0
3.       c0t3d0 <SUN1.05G cyl 2036 alt 2 hd 14 sec 72>
         /iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/
         ➥esp@5,8800000/sd3,0
Specify disk (enter its number): 0
selecting c0t0d0s0
[disk formatted]
```

From the Format menu we choose the Partition option, and from the Partition menu we choose the Print option. The Print option displays the partition table for the disk drive that we selected.

```
FORMAT MENU:
format> partition

PARTITION MENU:
partition> print

Current partition table (SUN2.1GB):
Part      Tag    Flag  Cylinders    Size     Blocks
  0       root    wm    0 -   40   30.43MB  (41/0/0)
  1       swap    wu   41 -  170   96.48MB  (130/0/0)
  2     backup    wu    0 - 2732    1.98GB  (2733/0/0)
  3 unassigned    wm    0      0            (0/0/0)
  4 unassigned    wm    0      0            (0/0/0)
  5 unassigned    wm    0      0            (0/0/0)
  6        usr    wm  171 - 2732    1.86GB  (2562/0/0)
  7 unassigned    wm    0      0            (0/0/0)
```

Specifying 1.98GB or 1980MB for a single partition on this drive, while following the usual `format` procedure, will fail with the error *1.98gb is out of range* or *1980mb is out of range*. Instead, you use a variation of the Modify option below to create a single partition that occupies the entire 1.98GB of available space.

```
partition> modify
Select partitioning base:
0. Current partition table (SUN2.1G)
1. All Free Hog
Choose base (enter number) [0]? 1
Do you wish to continue creating a new partition
table based on above table[yes]? return
Free Hog partition[6]? 0
Enter size of partition '0' [ 0b, 0c, 0mb]: 0
Enter size of partition '1' [ 0b, 0c, 0mb]: 0
Enter size of partition '3' [ 0b, 0c, 0mb]: 0
Enter size of partition '4' [ 0b, 0c, 0mb]: 0
Enter size of partition '5' [ 0b, 0c, 0mb]: 0
Enter size of partition '6' [ 0b, 0c, 0mb]: 0
Enter size of partition '7' [ 0b, 0c, 0mb]: 0
Okay to make this the current partition
table[yes]? [Enter]
Enter table name (remember quotes): '1.98GB'
Ready to label disk, continue? Yes
```

In this procedure, we use the All Free Hog option to tell `format` to grab all available space. This way, we don't have to worry about exactly what to type to use every byte on the disk drive. The Partition menu Print option reveals our new format.

```
partition> print
Current partition table (1.98GB):
Part  Tag         Flag    Cylinders   Size     Blocks
0     root        wm      0-2732      1.98GB   (2733/0/0)
1     swap        wu      0           0        (0/0/0)
2     backup      wu      0-2732      1.98GB   (2733/0/0)
3     unassigned  wm      0           0        (0/0/0)
4     unassigned  wm      0           0        (0/0/0)
5     unassigned  wm      0           0        (0/0/0)
6     usr         wm      0           0        (0/0/0)
7     unassigned  wm      0           0        (0/0/0)
```

We've completed the format procedure, so we can exit the program.

```
partition> quit
format> quit
```

Increasing storage space

Now that we have a new disk slice, we must prepare it for use by placing a new file system on it. You can use the `newfs` command to construct a new UFS (UNIX File System) file system.

```
# newfs /dev/rdsk/c0t0d0s0
newfs: construct a new file system
➥/dev/rdsk/c0t0d0s0 (y/n)? y
```

Before you do this, though, you should be aware that this step is the key to making as much disk space available to your application as possible. Just executing the `newfs` command without specifying any options will create a file system tuned for general usage, rather than the application you're building it for.

File-system details

In order to get the best use of your new file system for your application, you must understand how the disk usage pattern for your application interacts with the file system. Figure A shows a simplified file system, illustrating that a file system is partitioned into three sections: The *superblock* is a small disk area that describes the file system to Solaris, the *inodes* describe individual files on the file system, and the *blocks* are a collection of storage units in which the file system stores files.

When you create a file system, it has a fixed number of inodes. By default, Solaris creates one inode per 2K of storage for your file system. As you can see, there's a tradeoff. If your application uses only a few very large files, like a database, you'll waste a lot of disk space storing unused inodes. However, an application that uses very many small files, such as a news server, may run out of inodes long before using all the blocks in a file system.

When you create your file system, you also have control over the size of each block. When you use larger block sizes, access to larger files is more efficient. Solaris uses 8K by default, but you can change the default to 4K if you want. Thus, a file containing a single byte really consumes 4K or 8K of disk storage. While you might think it would be more efficient for a news server to use 4K blocks, read on....

Rather than blithely allowing the file system to waste so much disk space, the concept of a fragment allows the file system to reclaim some of that wasted space. If the last block in a file contains a lot of empty space, Solaris can use the empty space to store parts of other files. You can think of each block as a collection of up to eight sub-blocks. In the last block of a file (and only that last block), any unused sub-blocks may be used to hold data for another file.

Figure B shows three files that share a single fragmented block. The first file consumes three entire blocks and three fragments of the last block; the third file is so small, it completely fits in one fragment. The fragmented block still has two free fragments (in blue) available for use when one of these files is expanded or when the user creates a new file.

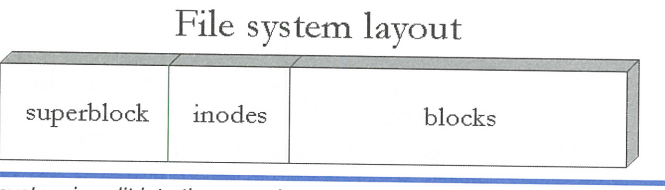
Solaris allows you to select among the following fragment sizes: 8K (if you're using 8K blocks), 4K, 2K, 1K, and 512 bytes (if you're using 4K blocks). This allows you to have 1, 2, 4, or 8 fragments per block. When you use fragments, Solaris operates more slowly. So you must decide whether the disk space is more important than speed for your system. We'll talk about speed again shortly.

Now let's think about our news server example. Would you select 4K or 8K blocks? With the increasing popularity of the Internet as a business tool, mail messages are getting larger every day. People are attaching spreadsheets, video clips, pictures, etc., to their messages. With the amount of header information on a mail message, most simple text messages are probably larger than 512 bytes. So it's probably not a good idea to use 512-byte fragments; they're probably too small.

In this case, we'd choose 1K fragments. Once you spend the time worrying about fragments, there's no difference between using a 4K block with 1K fragments or an 8K block with 1K fragments—except that larger files (such as the attachments we mentioned earlier) will work more efficiently with 8K blocks. Therefore, we'd choose 8K blocks with 1K fragments.

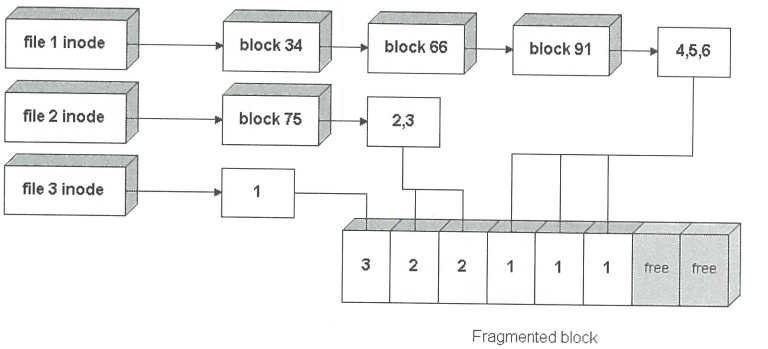
Two more important parameters for a file system are the optimization strategy and the amount of free space to reserve. When you're writing a new file to a file system, Solaris optimizes for time or space. For a database system, for instance, speed might be the single-most important parameter. For a news server, space may be. Solaris reserves some free space on the file system to allow it some room to perform housekeeping and because a file system operates more efficiently with more free space.

Figure A



A file system is split into three sections: superblock, inodes, and blocks.

Figure B



Solaris allows multiple files to share a single fragmented block to minimize wasted disk space.

Options for newfs

Now we're ready to get back to the heart of the matter. How are we going to set up our file system? Table A shows the tuning parameters you can use with the `newfs` command to optimize performance for your application.

Now that we know the basic operations of the file system and how our application uses disk space, we're ready to create our new file system. For example, let's suppose that we want to build a file system for a news server. We've already decided that we want to use 8K blocks with 1K fragments, so we know we'll use the switches `-b 8096 -f 1024`. Since most messages are short (somewhere in the

Table A

`newfs` Switches	Description
`-b` *size*	Sets the block size (4K or 8K)
`-f` *size*	Sets the fragment size (512, 1K, 2K, 4K, or 8K)
`-i` *bytes*	How many bytes of file system per inode
`-m` *percent*	How much free space to keep reserved
`-o` *opt*	Optimization method: speed or time

You can use these switches with `newfs` to customize your file system to the needs of your application.

neighborhood of 500 to 1,500 bytes), we could use the `-i 1024` switch to tell `newfs` that we want an inode for each 1K of space in our file system. This ought to give us enough inodes.

For a large disk drive (2.1GB certainly qualifies), you can get by with a lower percentage of free space, as Solaris will still have plenty of room for housekeeping. So rather than using the default value of 10 percent, we'll tell Solaris to leave only 5 percent free. To do so, we'll add the `-m 5` switch. Finally, since we're not so concerned about time, we'll tell `newfs` to optimize for space with the `-o space` switch.

Now we're ready to create our new file system:

```
# newfs -b 8096 -f 1024 -i 1024 -m 5 -o space
➥/dev/rdsk/c0t0d0s0
newfs: construct a new file system /dev/rdsk/
c0t0d0s0 (y/n)? y
```

Tuning your file system later

What happens if you make a mistake tuning your file system? Must you do a backup and rebuild the file system from scratch? If you want to change the number of inodes, or the block or fragment sizes, the answer is yes.

The good news is that Solaris provides a command, `tunefs`, that allows you to change the optimization strategy and amount of reserved space on the file system. For example, suppose we decided that reserving 5 percent of the file system wasn't giving us the performance we want. We could go back and change it to a higher value, say 7 percent, with `tunefs` like this:

```
# tunefs -m 7 /dev/dsk/c0t0d0s0
```

Just as in `newfs`, you use the `-m` option to set the percentage of space held back from normal users and the `-o` option to select the optimization strategy.

Conclusion

In many instances, you don't need to worry about the vagaries of tuning your file system. You just create your new file system and use it. However, some applications are very demanding of your file systems, and you want to maximize their performance.You'll probably want to tune databases for maximum speed, and you may want to squeeze out every last byte of storage for a news server. In this article, we've explained some of the basics of the file system so you can make informed choices.

Please keep in mind that you still need to measure system performance so you can fine-tune the system. After all, each modification in one subsystem will affect other systems as well. ❖

Disabling the automounter

By Sudhir Wadhwa

If you're running a small shop and have only a single Solaris computer, you might not appreciate the overhead of the automounter. Sure, it can manage users moving around the network and can simplify some NFS administration. But let's face it—with only a single computer, you may not want to spend any resources on the automounter.

In our shop, we simply wanted our users to be able to use the */home* directory. In addition, since we're not connected to other computers, except over the Internet, we didn't want the overhead of the automounter. However, in the default installation of Solaris, the automounter is active, controlling the */home* directory, and won't let you create an account there. If you try it, you'll get a nasty message like this one:

```
bash# mkdir /home/user1
mkdir: Failed to make directory "/home/user1";
Operation not applicable
```

The solution

If you want to put your user directories in */home*, the simplest method is to edit the

/etc/auto_master file and comment out the line in blue.

```
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid
/home           auto_home
/xfn            -xfn
```

Then you can reboot your computer, and you'll have complete control of your */home* directory. However, since you'll still have the overhead of the automounter, you should deactivate it to prevent it from consuming any resources, such as file handles, process table entries, and swap space.

You can disable the automounter by hiding the start and stop scripts that control the automounter in the */etc/rc?.d* directories. You can find the start and stop scripts that control the automounter with the command:

```
# grep -l automounter /etc/rc?.d/*
/etc/rc0.d/K69autofs
/etc/rc1.d/K68autofs
/etc/rc2.d/S74autofs
```

Now that you know the names of the scripts that start and stop the automounter, you can hide them by renaming them so that they don't start with a K or S. We prefer to add the prefix "hide_" to prevent Solaris from using the files during startup and shutdown:

```
# cd /etc
# mv rc0.d/K69autofs rc0.d/hide_K69autofs
# mv rc1.d/K68autofs rc1.d/hide_K68autofs
# mv rc2.d/S74autofs rc2.d/hide_S74autofs
```

Now when you reboot the computer, the automounter won't run at all. When you try to create your user account directories in */home*, it'll work just fine:

```
# mkdir /home/user1
```

When you connect your computer to a network, you want to take advantage of the automounter. Just move your user accounts from the */home* directory to */export/home*, restore the names of the start and stop scripts, and reboot. Be sure to export your */export* directory so other computers on the network can access any user directories on your computer!

Conclusion

While the features provided by the automounter are wonderful for networks, they're of no benefit to a standalone workstation. If you want to simplify administration of a standalone computer and provide as many resources as possible to the users, you may want to disable the automounter. ❖

CDE CUSTOMIZATION

Creating a new backdrop for CDE

Every so often, we all enjoy a change in scenery. It can be fun to take your notebook out in the backyard under the magnolia, sip lemonade, and work on some code. Or if your work environment isn't that flexible, maybe you can hang a new picture on your office wall. Ultimately, though, no matter what you do to your surroundings, you still find yourself staring at the same old scenery, as shown in Figure A on page 10.

Solaris provides many ways to customize the Common Desktop Environment (CDE). Using the Style Manager, you can change the color scheme, default fonts, and other features, but the default selection is meager. Fortunately, the CDE allows you to change just about everything. Unfortunately, it's not entirely clear how to change some things.

Boring backdrops

You've probably had a boring moment sometime back and viewed all the color schemes, fonts, and backdrops that the Style Manager offers. I know I have. I normally use the Pebbles backdrop on my main screen and Water Drops on my secondary screen, since I find these the best offered.

However, just because they're the best offered doesn't mean that I want to look at them all the time. As much as I liked Pebbles and

Water Drops the first time I saw them, I'm bored by them now. *I want variety!*

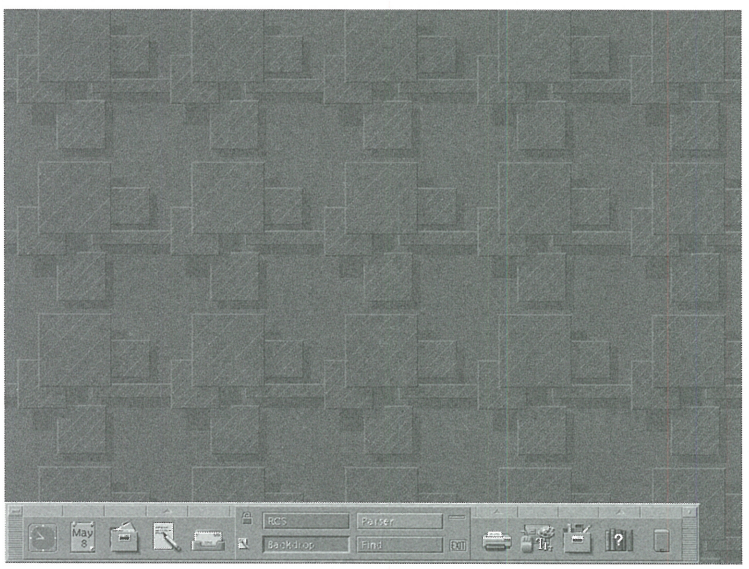
But how do you get new backdrops? It turns out that the procedure is fairly straightforward. First, you need a picture in the proper format. You place it in the appropriate location, then you restart the CDE. That's all there is to it!

The right picture...

The first thing you need is an appropriate picture for your backdrop. You can create a graphic in your favorite paint program, scan a photograph, render an image with POVRay, or download a file from the Internet. The only thing you must do is ensure that it's in the correct format. The CDE wants to display an XPM image. It'll ignore your images in other formats (such as TIFF or GIF).

So if your favorite picture is available only as a .JFIF image (i.e., JPEG File Interchange Format), are you out of luck? No! You can use the Image Viewer as a file format converter—it can read quite a few different formats. Some are very popular, such as GIF, TIFF, and JPEG, while others (Xerox Doodle Brush and Benet Yee Face File) are less so. The Image Viewer writes to considerably fewer formats, though it does write to the most popular ones. So all you should need to do is load the image into the Image Viewer and save it in the new format.

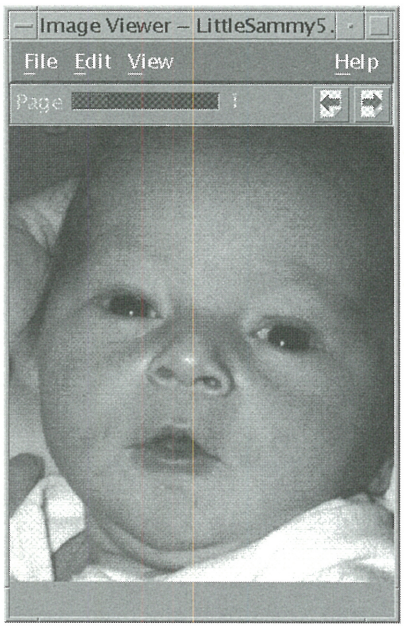
Figure A



Are you tired of the same old scenery?

Figure B shows our test image: A buddy sent me a picture of his new baby as a TIFF image. So using the Image Viewer, I loaded the image, scaled it down somewhat, then saved it as an XPM named *SamBranson.pm*.

Figure B



You can use the Image Viewer to convert an image to an XPM.

Please note that some picture file formats have multiple versions, and Image Viewer doesn't always handle them all correctly. For example, my first test picture was a 24-bit TIFF image, and it converted perfectly. However, another image created with a different program wouldn't work at all. The Image Viewer loaded it but wouldn't properly save it as an XPM image.

The right place...

Now that you have your picture, just place it in the */etc/dt/backdrops* directory if you want the image available to everyone and *$HOME/.dt/backdrops* if you don't. If no one on the system has created a new backdrop, then these directories won't exist, and you'll have to create the one(s) you want.

Once you place the image in the directory, you must restart `dtwm` by selecting Restart Workspace Manager... from the Workspace menu, which appears when you click the right mouse button on the desktop. Then start the Style Manager and select your new backdrop from the list, as shown in Figure C.

Hey, you're out of order!

As you select your new backdrop, you'll probably notice that all the backdrops Solaris provides are listed first in alphabetical order, while yours are unsorted at the end. If you find it annoying that Solaris' backdrops are in order and yours aren't, you can fix the problem.

The cause of the problem is that the Style Manager's Background selector box simply reads the files from each directory in the order they appear, rather than alphabetical order. Thus, if you want the files to be in alphabetical (or any other order), you must ensure that the files are placed in the directory in that same order. You can place them in alphabetical order using the following commands:

```
# cd /etc/dt
# mv backdrops bd
# mv bd/* backdrops
# rmdir bd
```

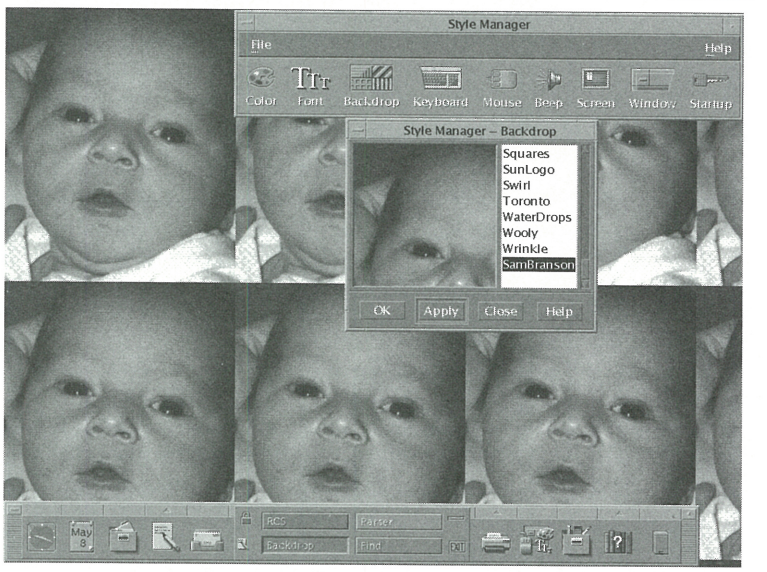
(If you'd prefer some other order, see the article "Changing the Order of Files in a Directory" on page 12.)

Please note that this method will only reorder the backdrops in the */etc/dt/backdrops* directory, so Solaris' backdrops will be in order at the front of the list, and yours will be sorted at the end of the list. (Also, if you put some images in *$HOME/.dt/backdrops*, you'll have *three* groups.) If you'd like *all* backdrops in order, you can put your backdrops in the */usr/dt/backdrops* directory, then sort that directory. However, Sun doesn't support this, and any customizations you make to this directory may be lost when you install the next patch or version of CDE.

Conclusion

Because of everyone's different aesthetics, the default setup for CDE can't be, nor is it intended to be, the best. It's simply a reasonable configuration you can use as a starting point for customizing your environment. Using the steps presented here, you should have no problem installing new backdrops when the ones Sun supplies start to bore you. ❖

Figure C



You now have a lively new backdrop for your CDE!

Solaris source code available to universities

If you've ever wanted to see what the code for Solaris looks like, you're in luck if you work for a university. Recently, Sun decided to make the source code for Solaris available to employees of universities and other educational institutions. Sun is offering the source code for the SPARC and Intel versions, including the DDK. Sun also includes E-mail support, discounted upgrade licenses, and lenient licensing terms with the offer. Unfortunately, those of us not in universities may only dream of seeing the source code.

The source code comes on a CD-ROM and costs only $100. (However, you can get it free if you order it when you order another Sun product!) The part number for this offer is UNSOL-2.5.1-SRC. For more information, check out *http://www.sun.com/edu/hot/source.html*.

Changing the order of files in a directory

Some programs process files in the order they occur in a directory, rather than sorting them before use. (The Style Manager's Background selector window is one example.) If you find it annoying enough, you can fix the problem. All you need to do is ensure that all the files in the directory are in the order that you want.

Since Solaris doesn't provide a command to swap around directory entries, the easiest way to do it is to clear all the files out of the *backdrops* directory, then move the files back into the directory in the order you want. The easy way to do this is to rename the directory, create a new directory with the original name, move the files, and delete the unneeded directory.

Suppose we want to put the files in */etc/dt/backdrops* in reverse-alphabetic order. First, we rename the directory to */etc/dt/bd* and create a new */etc/dt/backdrops* directory to ensure a clean directory structure:

```
# cd /etc/dt
# mv backdrops bd
# mkdir backdrops
```

Then we'll move the files from */etc/dt/bd* to */etc/dt/backdrops* in the order we want and delete the (now unnecessary) */etc/dt/bd* directory:

```
# mv bd/Tony.pm backdrops
# mv bd/LittleSam.pm backdrops
# mv bd/Jerry.pm backdrops
# mv bd/BigSam.pm backdrops
# rmdir bd
```

As you can see, this can be a tedious process. However, you can automate it simply enough. All you really need is a file containing the names of the backdrops you want, in the order you want them. You can use the following two commands to copy the files in the specified order. (For the purposes of this example, we're assuming that the sorted list is named *sort_list*, and it resides in the */etc/dt* directory.)

```
# cd /etc/dt
# for J in `cat sortlist`
> do mv bd/$J backdrops
> done
```

Using this method, you can place the files in your directory in any order you wish. However, if you just want your files in alphabetical order, you don't need the added complexity and effort of the `for` statement. You can sort all the directory entries in your directory by just using the `mv` command to move the files from the old directory to the new directory, like this:

```
# cd /etc/dt
# mv backdrops bd
# mv bd/* backdrops
# rmdir bd
```

This automatically alphabetizes your files because when the shell expands the argument `bd/*` to a list of filenames, it automatically sorts them.

It's not usually necessary to rearrange the order of files in a directory. Most programs either sort the files they work with, or the application doesn't care what order the files appear in. However, if directory order is important to you, you now have a method to enforce the directory order you want. ❖

Counting the number of files in a directory

Have you ever wanted to know how many files are in a specific directory? Without counting the directories, symbolic links, etc? We recently needed a simple way to count all the files in a directory, so we came up with the command pipeline shown here:

```
# ls -al $1 | grep ^- | wc -l
```

First, we use the `ls -al $1` command to create a list of all the files in the directory specified by `$1` (or the current directory if no directory is specified). Next, we pass the resulting output to the `grep ^-` command, which keeps only the lines that begin with a hyphen. (If you read the `man` page for `ls`, you'll see that all ordinary files begin with a hyphen—Any other character signifies that the directory entry is a directory, pipe, symbolic link, or other special file.) Finally, we use `wc -l` to count the number of lines, which is the number of files in the directory. QED.

Finding a program's process ID in a shell script

As you administer your system, sometimes you must find the process ID of a program. In past articles, we've shown you that you can find the process ID by using `grep` to search for the program name in a list of all the currently executing processes, as shown in Figure A.

Then, you select the second field of the appropriate line (the first one, in this case), and you have the process ID. When you're using this technique by hand, there's no problem. However, in a shell script, there are two problems. First, how do you know which line to use? Second, how do you get the second field out of the result?

Which line should we use?

In our example, `grep` finds and prints two processes: the process we care about (`sendmail`, in this case) and itself. When we execute it by hand, we just ignore the line containing `grep`. But in a script, getting the correct line requires a bit more work. One solution would be to reprocess the results with another `grep` command telling the second version of `grep` to print all lines not containing `grep`, as shown in Figure B.

Personally, we dislike this method because we run `grep` twice. In general, we want to be as efficient as possible in our shell scripts so we affect other users as little as possible.

The reason that `grep` finds itself is that the expression we're searching for happens to appear on `grep`'s command line. So if you can modify the expression enough, `grep` won't find itself. One way to change the expression is to surround the program name with quotes, then use a backslash to escape out a character in the middle of the program name, like this:

```
grep 'sen\dmail'
```

Alternatively, you could surround a character with square brackets, like this:

```
grep [s]endmail
```

In either case, we get the results we want—a single line of output from the `ps` command.

Getting the process ID

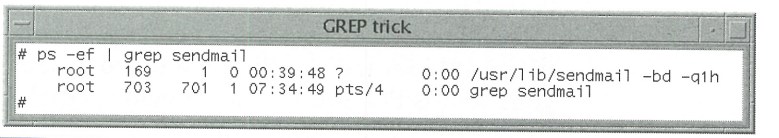
Now that we have the line we want, we simply extract the desired process ID. As usual, Solaris provides multiple ways to parse a line. If you're familiar with the `awk` command, you can extract the second field of input with the clause

```
awk '{ print $2 }'
```

Then you place the result in a variable for use in your shell. Figure C shows an example of this method. Now you have your process ID, neatly packaged in a variable!

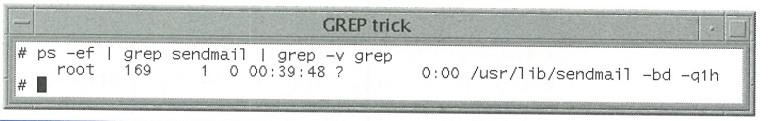
This works: It puts the process' ID into the variable PSID, but it's less efficient than it can be. Because `awk` is a flexible

Figure A



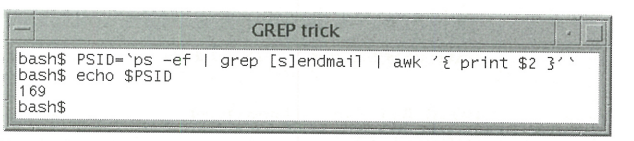
You can find the process ID of a currently executing program by using `grep` combined with `ps`.

Figure B



You can find exactly the process you want by adding another `grep` command to the pipeline.

Figure C



You can also extract a particular field using the `awk` command.

string-handling program, it's capable of taking over the search being done by the separate `grep` process. By preceding `awk`'s {...} construct with a regular expression specifying the line we want, we'll kill two birds with one stone:

```
$ PSID=`ps -ef | awk '/[s]endmail/ {print $2}'`
$ echo $PSID
169
```

Conclusion

Often, when we work through a process like this, it gives us some insight to the power of our own minds. Normally when we scan through the output of a command, we simply ignore anything we don't care about. While Solaris doesn't have our brain power, it will happily execute a few extra commands for us to find the data we want. ❖

SYSTEM INTEGRATION TIP

Configuring remote PPP dial-up client for MS Windows

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

Are you supporting a mixed bag of computing platforms and needing to implement PPP connectivity among the different platforms? With all the computers available, you'll probably see just about anything. Recently, we modified some laptops running Windows 3.1 to connect to our network.

In this article, we'll show you how to configure a Windows 3.1 PPP client to work with your Solaris PPP server. Two previous articles ("Configuring PPP under Solaris 2.x" and "Configuring Remote PPP Dial-up Client for Solaris 2.x," in the October 1995 and February 1996 issues) focused on configuring Solaris PPP servers and Solaris PPP clients.

On the Solaris PPP server side, we'll assume that you've implemented a PPP server with a dedicated dial-up telephone line. We'll also assume that your serial ports are set to 38,400bps, using 8 data bits, no parity, one stop bit, and hardware handshaking (RTS/CTS).

We'll also assume that you've entered IP addresses that conform to your site into your Solaris PPP server */etc/hosts* file. Here's a fragment of our */etc/hosts* file, with the IP addresses for PPP in blue:

```
127.0.0.1       localhost
157.21.3.40     worf
157.21.3.41     riker
157.21.3.42     troi
    •   •   •
157.21.3.50     ppp_server
157.21.3.51     ppp_client
```

Getting started

One of the easiest Windows 3.1 PPP clients to configure is a shareware program called Trumpet Winsock, available from Trumpet Software International.

In order for your Windows users to access the server, they'll need a PPP account. We tested our system by creating a PPP account on our server whose user name is *ppp_client* with the password of *ppptest*.

Configuring the PPP client

On the Windows client, the computers have either MS Windows v3.1 or v3.11 (Windows for Workgroups), and we've loaded Trumpet Winsock version 2.1B, along with the SLIPPER v1.5 packet driver by Peter R. Tattam, creator of Trumpet Winsock.

Start Trumpet Winsock and select *File/Setup*. You'll see the Trumpet Winsock Setup dialog box, shown in Figure A. (Be sure to use your own IP addresses!) Leave all parameters not altered in the steps below at their default settings.

First, you'll want to put the Solaris PPP server's ppp_client IP address into the IP Address field. (For our installation, we entered

157.21.3.51.) This tells Trumpet Winsock its address when connected to the Solaris PPP server. Since we're on a Class-C TCP/IP network, we entered 255.255.255.0 into the Netmask field.

Next, enter the Solaris PPP server's ppp_server IP address (157.21.3.50, in our case) in the Default Gateway field. This tells Trumpet Winsock the address of the computer that's acting as a gateway to the rest of the network.

Now we must tell Trumpet Winsock which computer will act as the Domain Name Server for the dialup computer. If your Solaris PPP host is set up as a DNS computer, then use its IP address. Otherwise, select the IP address of the computer that acts as the DNS server for your network. In the Domain Suffix field, enter the domain of your network to use when resolving names in the DNS. This field will support multiple domain suffixes.

The SLIPPER documentation recommends the following settings for PPP:

```
MTU 576, TCP RWIN 2048, TCP MSS 512
```

Then select Internal PPP and enter the SLIP port number, e.g., 2. This corresponds to COM2: on your Windows computer. (Don't let the mention of SLIP trouble you.)

Now set the baud rate (38400 for our installation), select Hardware Handshaking, and choose None as the Online Status Detection option. Save your changes, reboot Windows, and you're ready to test!

Manually testing your PPP connection

Select Dialer/Manual Login to initiate a PPP connection. Enter *ATDT* followed by the phone number of the Solaris PPP server. Respond to the Modem Login: prompt with *ppp_client*. Respond to the Password: prompt with *ppptest*. Don't be alarmed if random characters appear on the screen. Simply press the [Esc] key to stop the characters.

Once you're connected, minimize Trumpet Winsock and test your connection using your favorite Internet tools (such as Netscape, telnet, and ftp). When you're finished using the network, exit your Internet tool, and restore the minimized Trumpet Winsock application. Then press the [Esc] key again to get its attention, and type ATH and press [Enter] to terminate the PPP connection.

Automating your PPP connection

After you confirm the viability of your configuration, you can automate the login and logout procedures using two scripted command files called LOGIN.CMD and BYE.CMD. The command files were programmed using a scripting language specific to Trumpet Winsock. You can download the files germane to this article from the */pub/uploads/trumpet* directory at *riker.evms.edu* using anonymous ftp.

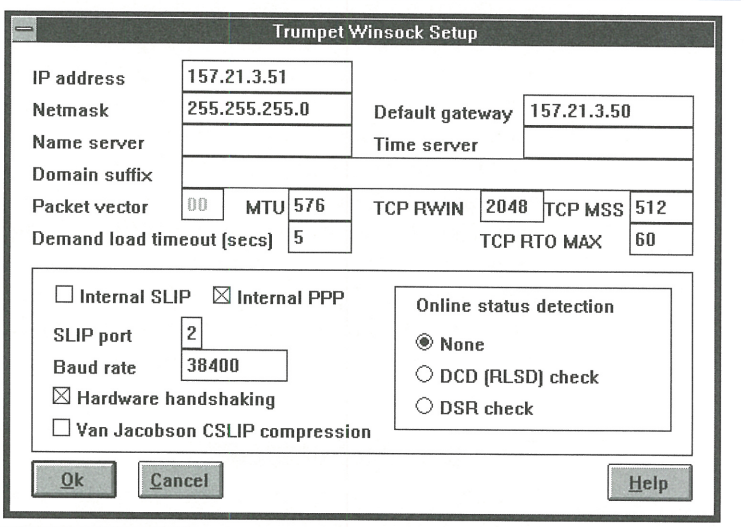
The LOGIN.CMD file allows you to preset a telephone number, user name, and password into the command file or to prompt the user for one or more of those entries. This command file also suppresses the random characters that appear during manual logins. You can customize and test the LOGIN.CMD and BYE.CMD command files from within Trumpet Winsock by selecting Dialer/Edit Scripts, editing the scripts, then selecting Dialer/Login and Dialer/Bye.

To test the command files, select Dialer/Login. LOGIN.CMD should establish a connection to the Solaris PPP server for you. Once you establish the connection, you're ready to use your Internet tools again. When you've finished, select Dialer/Bye to terminate the PPP connection.

Conclusion

It's unfortunate that some people must use primitive operating systems on their computers. However, you can make their lives brighter by enabling them to connect successfully to the rest of the world or at least your internal network. ❖

Figure A



You can configure Trumpet Winsock using the Setup dialog box.

PRODUCTIVITY TIP

Using audio alerts in your shell scripts

You just fire off a long procedure, then start taking care of some paperwork while your computer is chugging away. Isn't automation wonderful? About twenty minutes later, you check to see how far along your task is, only to find a blinking error message informing you of an input error, and your program hasn't even started.

Unfortunately, this situation is all too common. If you do it frequently enough, you become paranoid, checking your screen often to see if it's working, and if the program is making progress. Babysitting a program like this is counterproductive: You frequently stop what you're doing to glance at the screen, and you never dedicate yourself to the task at hand.

Consider this: You don't pick up your telephone throughout the day to see if there's someone on the other end. The phone makes some noise to alert you when it needs attention. You can do the same thing with your shell scripts: If your shell script detects an error, requires some data, or completes, it can play a sound to let you know. This way, you can work on your other tasks without constantly glancing at your monitor.

We use visual elements all the time when programming, but the problem with using only visual cues is that you must be looking at the screen to know that something's gone awry. If we add audio signals to our status messages, we have a better chance at detecting errors as they occur, rather than long after they occur.

The audioplay command

If you're going to play a sound on your workstation, you'll need the `audioplay` command, as well as some sound files to play. To do so, just enter the command:

```
$ audioplay sound_file.au
```

where *sound_file.au* is the name of the sound file you want to play. You can find sound files on the Internet, or you can record your own, using the `audiorecord` program. Just as your telephone service may offer different ring patterns for different members of your family, you can play different sounds for different events.

You should place your audio files in a consistent location to take advantage of the `AUDIOPATH` environment variable. This way, you don't have to specify the entire path name of the file to play it. Just set the `AUDIOPATH` variable to the list of directories to search for audio files.

Using with script files

You can use the `audioplay` command in your script files to let your script inform you when something bad (or good!) happens. If you use different sounds for different events, you can tell when certain events happen, even when you're not looking at your computer.

At our site, we typically use fun sounds, such as birds chirping, to alert us to expected events. For abnormal events, however, we custom record a message to tell us what's happened.

Conclusion

Many people dismiss audio in computers as a nuisance or suitable only for games. However, audio can be a useful tool to keep you appraised of what's going on. ❖