

# SINSIDE SOLARIS

Tips & techniques for users of SunSoft Solaris

## in this issue

- 1 Hiding data from prying eyes
- 3 Turning the File Manager into your command center
- 7 Processing command-line arguments simply in your shell scripts
- 10 A couple of ls tricks
- 11 Capturing a modal dialog box with Snapshot
- 13 Netscape Navigator available for Solaris Sparc and x86
- 14 Making fsck run faster on bootup
- 16 Add swap space to your computer

## Hiding data from prying eyes

If you work with sensitive data, you must find a way to restrict access to it. For example, if you worked in the Personnel department, you wouldn't want just anyone to be able to examine the payroll files. While you're undoubtedly familiar with using the `chmod` command to protect your files, it's tedious to use if you work with a large number of confidential files. In this article, we'll show you two unobtrusive techniques you can use to keep your secret data secret.

### Everything in its place

One simple way to maintain your data's privacy is to create a directory to hold all your secret files. Then you can use `chmod` to set the privilege mask for the directory to an appropriate level. This is a very simple technique to use, taking only a few minutes to execute.

For our example, we're going to create three directories in our home directory—*normal*, *secret*, and *topsecret*. For the *normal* directory, we won't change the permissions. For the *secret* directory, we'll remove access from everyone outside your group (i.e., other access), and for the *topsecret* directory, we'll remove all group and other access. Enter the following commands to create these directories:

```
$ mkdir normal
$ mkdir secret
$ chmod o= secret
$ mkdir topsecret
$ chmod g= topsecret
```

Now, only you (and, of course, root) can access the data files in your *topsecret* directory. Anyone in your primary group can access the data files in your *secret* directory. If people with an inappropriate group code attempt to use `ls` to examine your *secret* directory, they'll see the error message

```
$ ls -al topsecret
topsecret: Permission denied
total 2
```

Anyone who has access to your system can access the files in your *normal* directory, assuming, of course, that you created your *normal* directory with the normal permissions.

### The umask command

You saw the commands we used to create the directories. Using those commands, how could the *normal* directory be inaccessible? Have you ever wondered why, when you create a file, no one but you may write to it?

It turns out that when you create a file, Solaris examines your permissions mask to decide which permissions to give your file. You can change your permissions mask by using the `umask` command. You can also use `umask` to see which permissions Solaris will allow your file or directory to have when you create it, like this:

```
$ umask -S
u=rwx,g=rw,o=rwx
```

# 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](mailto: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](mailto:cr@merlin.cobb.zd.com)

## Staff

Editor-in-Chief ..... Marco C. Mason  
Contributing Editor ..... Al Alexander  
Production Artist ..... Liz Palmer  
Editors ..... Linda Recktenwald  
Karen S. Shields  
Circulation Manager ..... Mike Schroeder  
Editorial Director ..... Linda Baughman  
VP/Publisher ..... Lou Armstrong

## 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.

## Advertising

For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, ext. 430.

## Postmaster

Second class postage paid in Louisville, KY.  
Postmaster: Send address changes to:

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

## Copyright

© 1996, The Cobb Group. All rights reserved. *Inside Solaris* is an independent publication of The Cobb Group. The Cobb Group reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for personal and commercial use. Information furnished in this newsletter is believed to be accurate and reliable; however, no responsibility is assumed for inaccuracies or for the information's use.

The Cobb Group and its logo are registered trademarks of Ziff-Davis Publishing Company. *Inside Solaris* is a trademark of Ziff-Davis Publishing Company. Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, SunInstall, OpenBoot, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. UNIX and OPEN LOOK are registered trademarks of UNIX System Laboratories, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Notice that Solaris allows you to have all permissions for your files: r, w, and x. But when you create a simple file, it doesn't normally receive the x permission. Why is this? It's because Solaris uses the permissions mask to filter permissions in a file create request.

Normally, Solaris tries to create a text file with read and write permissions for you, your primary group, and all other interested users. In this case, Solaris will examine the permissions you requested and discard any that don't appear in the permissions mask. Since the permissions mask doesn't allow write access to the group and others, those permissions won't be granted. Thus, your file will have read and write access for you and read only access for your group and all others.

Similarly, when Solaris creates a directory, it tries to grant read, write, and execute privileges to everyone. (You need the execute permission on a directory to be able to cd to the directory.) However, your permissions mask prevents write access to your group and all other users; everyone will have read and execute privileges, but you alone will have the write privilege. (Thus, you'll be the only person allowed to create and delete files in your directory.)

If you're always working with highly sensitive data, you may

want to set your default permissions mask to prevent your primary group, all other users, or both from accessing your files. For example, if you want to remove all rights for all users outside your primary group, you can use the command

```
$ umask -S 0=
```

Please note that we've been using the -S option on umask. This allows us to specify the permissions symbolically. If you don't mind working with octal numbers, you can omit the -S and work with the permissions as octal numbers. For more details, read `man umask`.

## Conclusion

The two simple methods we described can help you keep your files secure. Be aware, however, that your files are only as secure as your password. Don't spend a lot of your effort making your files and directories secure before taking steps to keep your password private.

For this reason, you'll want to be careful about which files are for your use only and which you allow other members of your group to access. If any group members are lax about keeping their own passwords safe, this carelessness could compromise your data. ❖

## SunSoft unveils Internet tools

SunSoft recently announced the Solaris Internet Access PlusPack, a suite of tools that make Internet access and authoring simpler and cheaper. It bundles together:

- Java-based Desktop browser
- Netscape Navigator 2.011
- HoTMetaL Light 2.0
- Video and Audio tools
- Sun's Java Virtual Machine

This product costs only \$99. It runs on Solaris v2.4 or later machines using Sparc or x86 architecture. It runs under both CDE and OpenWindows. You can get more information about this product by calling 1-800-SUNSOFT (follow prompt 1), or by visiting the Web page <http://www.sun.com/Solaris/products>.

# Turning the File Manager into your command center

By Al Alexander

**M**any System administrators make a basic mistake about the users of their systems. They believe that most users enjoy working with the arcane intricacies of Solaris as much as they themselves do. It often comes as a shock when they discover that some users are more interested in completing their work than in working on the computer. These users rightly believe that their other work is more important than learning the syntax and command-line options of various Solaris commands.

If you're in this situation, you can become an overnight hero when you learn this lesson. Instead of forcing your users to learn the command-line arcana of Solaris, you can find an easier way to bring the power of UNIX to them. One way you can do this is by transforming the seemingly innocent File Manager into a powerful, flexible command center.

Once you make this change, your users will have the ability to run a wide variety of UNIX commands from the File Manager without worrying about any obscure calling conventions. They can, for example, run their CAD and finite-element analysis (FEA) applications, run personal tape backups, manipulate custom batch queues, etc.—all from the confines of the File Manager.

This relatively low-technology solution brings home an important lesson in system administration: No matter how cool and interesting you find the new technologies, the users you support often don't care. It's not vital that you solve every problem with a flashy application. To most users, a simple, reliable way to do their jobs is what's important. In other words, they're looking for a way to make things work better so they'll be more productive.

In this article, we'll show you how you can extend the File Manager by adding new commands. This allows your system's users to do all their work from within the File Manager. Your endeavors may endear you to your users enough so that you receive that coveted "Most

Valuable Employee of the Month" award you've been wanting.

## Why use the File Manager?

The beauty of the File Manager approach is that for most end users, using a computer system is a file-oriented endeavor. All day long, users open, close, create, copy, delete, modify, and print files (or documents, if you prefer).

If only you could make the File Manager perform other tasks, you'd have the perfect file-oriented tool for your users. Wouldn't it be great if you could simply select a set of files, select Fax, and have the File Manager ask you for the fax number to send the files to?

Obviously, we wouldn't tantalize you without telling you how to accomplish the task, so let's get down to business. First, you need to know that this trick works with the default File Manager used by OpenWindows, but not with the CDE version of File Manager.

Please note that if your users run CDE, then you can customize their environment in other ways to do similar tasks. However, we won't cover those techniques in this article. You can still use this technique, if you want, by running the OpenWindows version of File Manager under CDE. To do so, you should execute the program `/usr/openwin/bin/filemgr`.

## Locating the custom commands menu

Solaris' File Manager for OpenWindows lets you create your own commands with the support of the Custom Commands option. Once you've defined them, custom commands are available to users through the File menu. You can see this in operation by starting the File Manager, clicking the File menu button, and then selecting the Custom Commands option. The resulting list shows you which custom commands you currently have configured for your system. The default application available from the custom commands menu is a UNIX Shell.

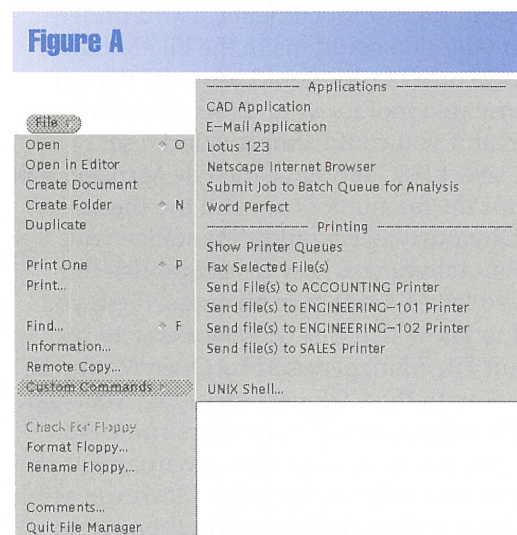
In [Figure A](#), we've created a sample menu to show you some of the things you can do with the Custom Commands option. As you

see, you can add a large variety of commands. Please note that the UNIX Shell... entry is always on the Custom Command menu. If you invoke this option, it simply opens a Shell Tool window for you. Now let's see how to create our own custom commands.

## Adding a custom commands

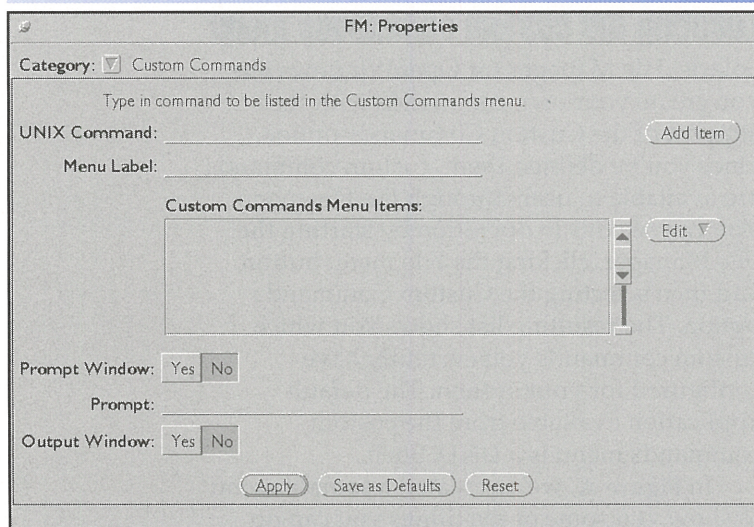
It's very simple to add custom commands to the File Manager. We'll demonstrate the process by creating a command that will allow users to print one or more files to a printer named Accounting.

To begin, start the File Manager. Then click the Edit button, and select Properties... from the Edit menu. When you do so, you'll see the FM: Properties dialog box. At the top



This sample menu shows some of the flexibility available when you use the Custom Commands option.

Figure B



You can create new commands for File Manager with the FM: Properties dialog box.

of the dialog box, you'll see a dropdown menu labelled Categories. Click the down arrow and select Custom Commands. When you do so, the FM: Properties dialog box should look like the one shown in Figure B.

You'll use this screen to define all your new custom commands. Next, we'll describe all the items in this dialog box and simultaneously build our new custom print command.

## Choosing a command

The first field on the Custom Command section of the FM: Properties dialog box is the UNIX Command field. You use this field to enter the command you want the custom command to execute. You may enter any command or script file in this field, along with any parameters you want to use.

Let's begin with our new print command. Normally, when you want to print files to the Accounting printer, you execute the command

```
lp -dAccounting FileList
```

replacing FileList with the list of files you want to print. Similarly, if you want to print on a special form, you'd add the form name with the -f option. That's simple enough. But where do we get the list of files?

## The \$FILE token

The File Manager defines a special token, named \$FILE, that you use to access the list of currently selected files. To do so, you simply add the \$FILE token in the UNIX Command field in the position where you want the file list to appear. When the File Manager executes your custom command, it will replace the \$FILE token with the entire list of files you've selected.

## The \$ARG token

Please note that the File Manager provides another token for your custom commands: the \$ARG token. The File Manager allows you to prompt the user for special options for your custom command, as we'll discuss in the "Prompting the user" section. When you use the \$ARG token, the File Manager replaces it with the user's response before it executes the custom command.

For our new print command, we'll simply replace FileList with \$FILE. Additionally, since the user may want some special print options, we'll add the \$ARG token to allow this. Let's get started. Click on the UNIX Command field to select it. Now type the command you

want to execute. Since we want our custom command to print a file to the Accounting printer, with the possibility of incorporating some options, we'll enter the command

```
lp $ARG -dAccounting $FILE
```

Now suppose the user highlighted two files named *list.c* and *list.h* and then invoked this custom command. Let's also suppose that the File Manager prompted the user for extra options, and the user replied *-fLEGAL*. In this case, the File Manager would replace the *\$FILE* token with *list.c list.h*, then replace the *\$ARG* token with *-fLEGAL*, and execute the command as follows:

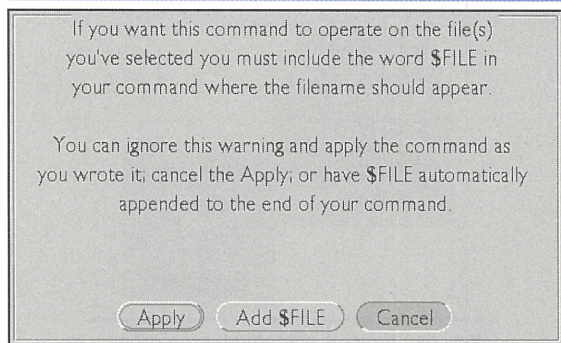
```
lp -fLEGAL -dAccounting list.c list.h
```

Remember that the *\$FILE* and *\$ARG* tokens are case-sensitive. If you use any lowercase letters, the File Manager won't recognize them. For example, if you use *\$File* instead of *\$FILE*, then the custom command we set up will try to print a file named *\$File*, no matter what files you've selected.

Since the File Manager is file-oriented, it expects custom commands to have an embedded *\$FILE* token. If you haven't placed the *\$FILE* token in your custom command, or if you've misspelled it, the File Manager will alert you with the message box shown in [Figure C](#) when you click the Add Item button.

If you don't intend for your custom command to use the *\$FILE* token, just press the Apply button. If you want the File Manager to add *\$FILE* to the end of your custom command, press the Add *\$FILE* button. Otherwise, press the Cancel button and fix any problems with your custom command.

**Figure C**



If you create a custom command without including the *\$FILE* token, the File Manager alerts you with this dialog box.

## Naming your custom command

Now that we've told the File Manager what the custom command should do, we need to give it a label so the user can find it. The File Manager provides the Menu Label field to allow you to give your custom command a name. To do this, click on the Menu Label field and enter the command name; in this case, we'll enter *Print to Accounting*.

## Prompting the user

While discussing the *\$ARG* token, we alluded to the fact that you can have the File Manager prompt the user for more information when the user executes the custom command. You can use this feature by setting the Prompt Window field to Yes, then entering the prompt text you want the user to see in the Prompt field.

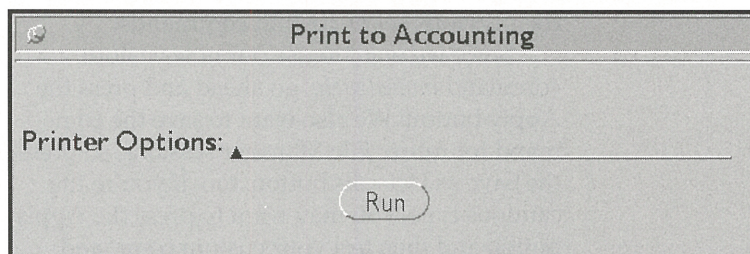
When the user executes a custom command that has a prompt, the File Manager will create a new window to prompt the user with your custom prompt string. Then the File Manager will replace the *\$ARG* token with the text you enter and the *\$FILE* token (if present) with the list of selected files. Finally, the File Manager will execute the custom command.

In our example, we used the *\$ARG* token in our custom command, so we should prompt the user for some additional information. To do so, set the Prompt Window field to Yes, then in the Prompt field, enter *Printer Options* to allow the user to specify any additional printer options for the job. When the user executes this custom command, the File Manager will present the Print to Accounting dialog box shown in [Figure D](#).

## The output window

For many of your custom commands, you may choose to display the output of the UNIX command. You can do so by setting the

**Figure D**



Your custom commands can tell the File Manager to prompt the user for further information on a command.

Output Window field to Yes in the File Manager Properties dialog box. Then, any output from the UNIX command will appear in a separate window that resembles a Shell Tool window. To close the output window, you select it (typically by clicking the mouse inside the window) and press the [Return] key.

For our example command, we don't use the output window. So, click the No button next to the Output Window field to ensure that we don't accidentally open one.

However, when you're creating a custom command and it's giving you trouble, you may want to use the output window as a debugging aid. Many UNIX commands emit error messages when they fail. If you turn on the output window, you'll be able to see any error messages your custom command generates when it tries to run. So remember, if your custom command fails, turn on the output window and examine any output it may provide.

## Adding your custom command to the menu

When you create your custom command, you need to add it to the Custom Commands Menu Items box. To do so, click the Add Item button along the right side of the dialog box.

Adding the custom command to the Custom Commands Menu Items box doesn't tell the File Manager to begin using your new custom command. If you want to use these custom commands in the current File Manager session, you must press the Apply button, which tells the File Manager to forget any custom commands it previously stored and load new ones from the Custom Commands Menu Items box.

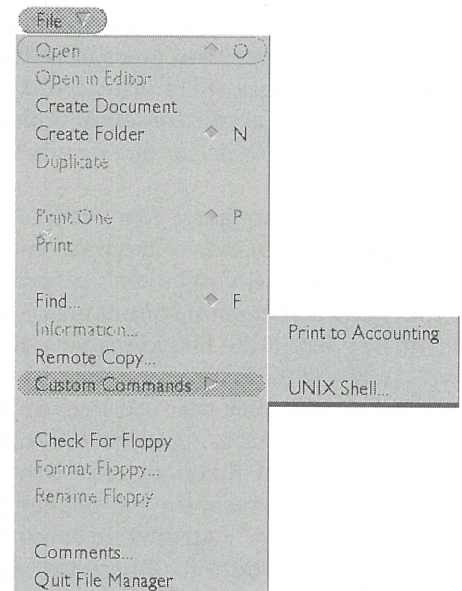
If you click the Save as Defaults button, you're telling the File Manager to save these custom commands in a special file. From now on, each time you start the File Manager, it will load your custom commands. If you don't click the Save as Defaults button, then the next time you start the File Manager, it will reuse the previous set of custom commands.

Since we want to use our new custom command *immediately*, go ahead and press the Apply button. We also want to save the command for future File Manager sessions, so press the Save as Defaults button, too. If you're the cautious type, you may want to press the Apply button and then test your custom command before actually saving it permanently.

At this point, your new print command should now be available from the Custom Commands submenu on the File menu. To check this, click the File button to pull down the File menu, then select the Custom Commands option. When you do so, you should see the menu label Print to Accounting, as shown in Figure E.

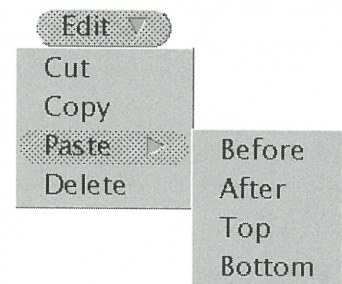
To use your new command, first select a file (or files), then select the Print to Accounting item. When you do so, the File Manager will prompt you with the dialog box shown in Figure D. Once you respond and press the Run button, File Manager will send the selected files with the specified options to the Accounting printer.

Figure E



As you can see, the Custom Commands submenu now has our Print to Accounting item available.

Figure F



You can remove and change the order of the Custom Commands menu with the Edit pulldown menu.

## Changing your custom command list

Take another look at Figure B. See the Edit button to the right of the Custom Commands Menu Items list box? This pulldown menu allows you to delete and rearrange items in the Custom Commands Menu Items list box. This menu provides the options Cut, Copy, Paste, and Delete. The Paste submenu provides four variations: Before, After, Top, and Bottom, as you can see in Figure F. You can use these options just as you'd imagine. The Paste submenu lets you decide where to paste any custom command.

**BUG ALERT:** Please note that you should use these edit options sparingly. It was our experience on all the systems on which we tried these options that the File Manager would easily crash if we overused these options.

## Conclusion

Creating custom commands lets you turn your File Manager into a useful command center for your users. Once modified, the File Manager can become the end users' primary interface to the computer system and network, allowing them to launch UNIX commands, custom utilities, and full-blown applications from one location. ❖

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

## SHELL SCRIPT TECHNIQUE

# Processing command-line arguments simply in your shell scripts

By Al Alexander and Marco C. Mason

**Y**ou can describe one of the fundamental principles of UNIX as the ability to build on the foundations laid by other users. If you need to write a shell script to do a job, you can often rely on programs that others have written to do at least some of the work for you.

One of the popular conventions of UNIX programs is that you can modify their behavior by providing command-line options. In order to make things simpler for users, UNIX follows the convention that all options should precede any arguments affected by them. In addition, these arguments can occur in any order, and they can be specified in one clump, separately, or somewhere in between. Thus, the following command lines are all equivalent:

```
myprog -l -v -t 60
myprog -t 60 -vl
myprog -vlt 60
```

## Enter getopt

One of the programmers who went before us created the `getopts` command. If you enjoy writing shell program utilities that run from

the command line and you haven't used the `getopts` command, you'll soon find that it will become an indispensable tool for your Bourne shell programs. The `getopts` command does nearly all the work required to parse command-line options.

If you've ever tried to write a shell script that accepts multiple command-line options using the standard UNIX conventions, you've no doubt found it a daunting task. With the `getopts` command, you can make short work of that task and concentrate your time and effort where it matters: making the shell script do its job.

## How do you use getopt?

The `getopts` command isn't very difficult to use. You call it with the syntax

```
getopts argList var
```

where `argList` is a list of command-line options that you want `getopts` to parse, and `var` is the shell variable you choose to hold the next option. Each time you call `getopts`, it searches the command line for the next option. When it finds

one, it places the option name in *var* and returns a 0. When it finds no more options, it returns a nonzero value. Because of this convenient behavior, `getopts` lends itself well to being used in a loop to process all the options at once, as shown in the shell script in [Figure A](#).

The `argList` parameter tells `getopts` that it should search for the options `-a`, `-h`, `-l`, and `-v`. If it finds any of these, it sets the value of the `cmdLineOption` variable to the option it found and returns 0. This loop then uses the `case` statement to perform specific processing based on the option that was found. When `getopts` finds no more options, the loop terminates. Here's what the shell script does when you run it:

```
$ ./myprog -al -h
ALL is TRUE
LONG is TRUE
HELP is TRUE
```

As you can see, you can place the options in any order on the command line, and `getopts` processes them in that same order. Also, `getopts` enforces no specific ordering criteria on `argList`. We put the options in alphabetical order so it's easier to see which ones are acceptable to the shell script.

All of this is just fine when you're processing options that are specified by a single letter. But what about options that specify values?

## Even more flexibility

Many times, you'll want a user to be able to include a value after a command-line argument. For instance, many UNIX commands use the `-t` command-line argument to represent a time value and use `-n` to represent a

numeric value. It happens that the `getopts` command supports options with values in a straightforward fashion.

When you want `getopts` to handle an option that accepts a parameter, you tell it to do so by putting a colon (:) after the specific option letter in the `argList` parameter. When `getopts` discovers that an option you've specified takes an argument, it reads the argument and places it in the shell variable `OPTARG`.

We've modified the shell script in [Figure A](#) to accept two additional options, `-n` and `-t`, which take parameters. We've highlighted the changes by showing the additions in blue in [Figure B](#).

Notice the changes in both the `getopts` command and the `case` statement. In the `getopts` command, we added the two possible options `-n` and `-t` followed by a colon, just as you'd expect. In the `case` statement, we added two new cases, one for each new option. As you can see, when we process an option that takes an argument, we copy the value of the `OPTARG` variable into a new variable. By doing so, we don't lose the value when we process the next argument. With the new version of *myprog* shown in [Figure B](#), all of the following commands will succeed:

```
myprog -a -l -n 2 -t 60
myprog -alvn 2
myprog -t 60 -n 5 -alv
myprog -t60 -n5 -alv
myprog -h
```

**Figure A**

```
ALL=FALSE
HELP=FALSE
LONG=FALSE
VERBOSE=FALSE

while getopts ahlv cmdLineOption
do
  case "$cmdLineOption"
  in
    a) ALL=TRUE;      echo "ALL is TRUE";;
    h) HELP=TRUE;    echo "HELP is TRUE";;
    l) LONG=TRUE;    echo "LONG is TRUE";;
    v) VERBOSE=TRUE; echo "VERBOSE is TRUE";;
    esac
  done
  shift `expr $OPTIND - 1`
```

Here's a simple example of a program that uses `getopts` to help parse the argument list and process options.

**Figure B**

```
ALL=FALSE
HELP=FALSE
LONG=FALSE
VERBOSE=FALSE
NUMBER=0
TIME=0

while getopts ahln:t:v cmdLineOption
do
  case "$cmdLineOption"
  in
    a) ALL=TRUE;      echo "ALL is TRUE";;
    h) HELP=TRUE;    echo "HELP is TRUE";;
    l) LONG=TRUE;    echo "LONG is TRUE";;
    n) NUMBER=$OPTARG; echo "NUMBER is
    ↪ $NUMBER";;
    t) TIME=$OPTARG;  echo "TIME is $TIME";;
    v) VERBOSE=TRUE; echo "VERBOSE is
    TRUE";;
    esac
  done
  shift `expr $OPTIND - 1`
```

In this version of *myprog*, we've added the ability to accept two new options that take parameters.



Wow! The `getopts` command provides tremendous power with very little programming effort on your part. Not only can you specify command-line options in any order, but you can process options that accept arguments.

## Error checking

But wait! What happens to our program if we specify an option that doesn't exist? Let's try it by giving `myprog` a couple of bad options:

```
$ ./myprog -axl -y
ALL is TRUE
./myprog[8]: getopts: x bad option(s)
./myprog[8]: getopts: y bad option(s)
```

Well, *that* explains why the error messages from some UNIX commands are so cryptic. Many UNIX utilities tell you *what* failed but not *why* they failed. The tool does its best, telling you its name (`getopts`) and what was wrong (`x` and `y` are bad options), but it can do more than that. If you don't go the extra mile and put in some better error-handling code, then you can expect cryptic error messages like these.

The aforementioned error message simply tells us that the shell found an error in `myprog` on line 8 and that the error occurred because `getopts` reported `-x` and `-y` as bad options. While this may be an acceptable error message for a programmer or system administrator, you can rest assured that an ugly message like this will net you several phone calls from your users.

So what can we do about this situation? When `getopts` can't process an option, it sets `var` to `?` so your shell script can tell that the user entered an erroneous option. Also, if you start `argList` with a colon, `getopts` will place the unrecognized option character in the shell variable `OPTARG` and won't print an error message.

Now let's add some better error handling to our script. [Figure C](#) holds our new script. Again, we've highlighted the changes in blue so you can see them more easily.

The first section of our shell script highlighted in blue sets the value of two variables: `ABORT` and `UNREC`. The `ABORT` variable tells the shell script whether to terminate and print an error message after we process the parameters. We clear the `UNREC` variable, since we use it to build a list of all unknown options we find during processing.

The next new section of code processes the unknown options. Since `getopts` sets `var` to `?` for each unknown option, we can use a

single clause in our `case` statement for processing. Here, we simply add the unknown option to `UNREC` and set `ABORT` to tell the script to print an error and quit.

The final new section of code tests the `ABORT` variable. If the `ABORT` variable is set, we print a message describing the proper use of the program, and the list of unknown arguments found. Now, if we run `myprog` with the same argument list, we get a much better error message.

```
$ ./myprog -axl -y
ALL is TRUE
usage: myprog [-a] [-h] [-l] [-n num] [-t time] [-v]
where: a=ALL files, h=enable HELP, l=LONG output
       n=count, t=start time, v=VERBOSE output
Bad argument(s) found: x, y
```

Figure C

```
ALL=FALSE
HELP=FALSE
LONG=FALSE
VERBOSE=FALSE
NUMBER=0
TIME=0
ABORT=0
UNREC=

while getopts :ahln:t:v cmdLineOption
do
  case "$cmdLineOption"
  in
    a) ALL=TRUE;      echo "ALL is TRUE";;
    h) HELP=TRUE;    echo "HELP is TRUE";;
    l) LONG=TRUE;    echo "LONG is TRUE";;
    n) NUMBER=$OPTARG; echo "NUMBER is $NUMBER";;
    t) TIME=$OPTARG;  echo "TIME is $TIME";;
    v) VERBOSE=TRUE;  echo "VERBOSE is TRUE";;
    ?) if [ ABORT -ne 0 ]; then
        UNREC="$UNREC, "
      fi
        UNREC=$UNREC$OPTARG
        ABORT=1;;
  esac
done
shift `expr $OPTIND - 1`
if [ ABORT -ne 0 ]; then
  echo "usage: myprog [-a] [-h] [-l] [-n num] [-t time] [-v]"
  echo "where: a=ALL files, h=enable HELP, l=LONG output"
  echo "       n=count, t=start time, v=VERBOSE output"
  echo "Bad argument(s) found: $UNREC"
  exit
fi
```

The final version of `myprog` now has both error checking and reporting.

Please note that the `-l` option was never processed in either this or the previous version. The reason for this is that some options may take arguments, and you're allowed to put an argument that takes parameters at the end of a cluster of options. Therefore, `getopts` can't be sure of the validity of the characters following the bad option. Thus, it ignores them.

## Conclusion

The `getopts` command is a terrific tool for processing command-line arguments from within shell programs. Using the code we developed in this article as a template, you can almost instantly and painlessly add very powerful command-line arguments to all of your shell programs. ❖

## COOL TOOL TRICKS

# A couple of ls tricks

By Al Alexander

On occasion I hear people report problems when they use the `ls` command with directory listings. Sometimes they get unexpected results. In this article, we offer two quick tips for using the `ls` command with directories—and getting the proper results.

## List only the directories

At times you'll want to generate a list of all the directories contained within the current directory. To be more precise, you want to see a listing of directories only—no other files.

There are several ways to do this. One quick way to generate a long list of all of the directories in the current directory is to use the command

```
ls -al | grep '^d'
```

This line tells Solaris to create a list of all the files in the current directory. The `-l` switch tells `ls` to print the directory in long format. This is important because we're going to examine the first character in the permissions mask to decide whether the file is a directory. The `grep` command goes through the directory listing, line by line, and prints each line that begins with the letter `d`.

As you may remember from last month's article "Hard and Soft File Links," when you create a symbolic link to a directory, the `ls -al` command will show an `l` in the first position, rather than a `d`, even if it's a directory. The `-L` switch on `ls` tells `ls` that in the case of a symbolic link, to print the permissions for the file linked to, rather than the permissions for the link itself. This avoids confusion, since a symbolically linked directory will show a `d` in the first position, rather than an `l`.

You can see some of the output of this command, run in the root directory of my Solaris 2.5 computer, in Figure A. Please note that the `/bin` directory, which is implemented as a symbolic link to the `/usr/bin` directory, appears normal (i.e., with a `d` permission) here.

If you just want to print the names of the directories, but not the long display, simply add a `cut` command at the end of your command chain. Since `ls` prints the filenames at column 55 with the `-l` format, we can just add the command `cut -c55-` to remove all the other information on the line. Thus, our revised command looks like this:

```
ls -al | grep '^d' | cut -c55-
```

Figure A

```
$ ls -al | grep '^d'
drwxr-xr-x 30 root   root   1024 Jun 19 00:09 .
drwxr-xr-x 30 root   root   1024 Jun 19 00:09 ..
drwxr-xr-x 12 root   other   512 Jun 19 00:11 .dt
drwx----- 6 root   other   512 Apr  4 21:36 .fm
drwxr-xr-x 2 root   other   512 Apr  2 10:28 .wastebasket
drwx----- 4 root   other   512 Apr 16 14:15 Mail
drwxr-xr-x 2 root   root    512 Apr  1 22:36 TT_DB
drwxr-xr-x 2 root   other   512 May 10 11:51 a
drwxrwxr-x 2 root   bin     7168 Apr 29 18:38 bin
drwxr-xr-x 3 root   nobody  512 Apr 24 09:14 cdrom
drwxrwxr-x 17 root   sys    3072 Jun 16 15:22 dev
drwxrwxr-x 5 root   sys    512 Apr  1 21:01 devices
drwxrwxr-x 25 root   sys    3072 Jun 19 10:20 etc
```

With a little effort, you can generate a list of only the directories within any directory.

On the other hand, if you don't mind a forward slash at the end of each directory name, here's a simpler way to accomplish the same task:

```
ls -alp | grep '/'
```

This command will print the same information. If you want only the directory names, omit the `-l` option for `ls`. The `-p` option of `ls` is the key to this method. It tells `ls` to put a slash (/) on the end of each filename specifying a directory. Then you tell `grep` to print all lines that end with a slash.

## Listing a directory's properties, but not its contents

Let's discuss another question we encounter frequently: How do you look at the permissions, ownership, or size of a directory, rather than its contents? Your first instinct might be to type

```
ls -l /etc
```

However, if you ponder it a bit, you'll notice that you're actually asking `ls` for a list of all the information about the *files* in the directory, rather than about the directory. Users frequently forget about the `-d` option of the `ls` command, which tells `ls` to give us the permissions of the directory file itself, rather than to list the directory's contents. Thus, what you really need to type is

```
ls -ld /etc
```

## Conclusion

As you can see, it's easy to get a list of all the subdirectories in a directory. As is often the case with UNIX, there is usually more than one way to do a job. It's also easy to forget things you don't do every day, so it's no wonder people forget the `-d` switch on the `ls` command. ❖

## DOCUMENTATION TIP

# Capturing a modal dialog box with Snapshot

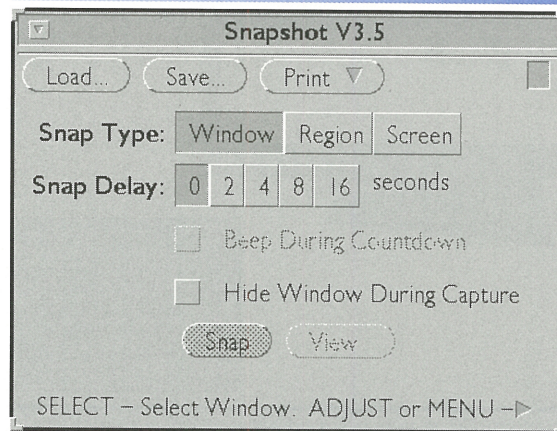
If you write very much system documentation, you probably use the Snapshot tool to grab screen images to illustrate your procedures. For most screens, it's a straightforward procedure. However, some dialog boxes under Solaris are modal, meaning that the desktop freezes until you answer it. This delay prevents you from switching to the Snapshot tool to capture the image. As it turns out, you can use a relatively simple method to maneuver around this shortcoming.

Normally, when you start the Snapshot tool, Snapshot's control panel looks like [Figure A](#). Using these default settings, to snap a window, you simply press the Snap button and select the window you want to capture. Then the Snapshot tool grabs it.

A good example of a modal dialog box is the prompt the File Manager uses when you try to quit. Once you select the Quit File Manager option from the File menu, File Manager presents you with a dialog box asking you to

confirm that you really want to quit. While this dialog box is active, you can't switch to the Snapshot tool to capture the image. What can we do to remedy this situation?

Figure A



When Snapshot starts, it's already set up for the simple way to capture images of application windows.

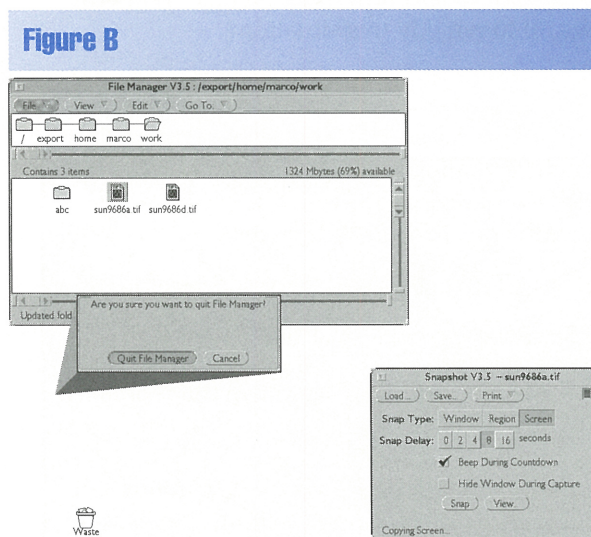
## How to do it

In order to take a picture of a modal dialog box, we have to cheat a little bit. First, rather than trying to capture a window, you'll tell Snapshot that you want to capture the whole screen. Next, set the Snap Delay time value to one large enough to allow you to get the modal dialog box on the screen. (We usually use 8 seconds). Then turn on the Beep During Countdown field, so you can keep track of your time.

Now you're ready to capture your image. First, prepare your application so that you only have to perform one operation to invoke the modal dialog box. Then switch to the Snapshot window and press the Snap button. You'll have several seconds to switch back to your application and invoke the modal dialog box. While you're working, you'll hear Snapshot beep every second to let you know that your time is running out.

Please note that once the modal dialog box appears on screen, the countdown beep stops. This is because the modal dialog box prevents other screen I/O from occurring while the modal dialog box is active. Now wait until you're sure that the Snap Delay time is over. Once you're sure, close the dialog box. When you do so, Snapshot will emit a burst of beeps telling you that it saved the image. Please note that your screen shot will show in its depressed state the button you must click to close the dialog box.


Now you have an image of the whole screen, as shown in [Figure B](#). However, you probably only want the dialog box.




The first step in capturing a modal dialog box is to grab the entire screen.

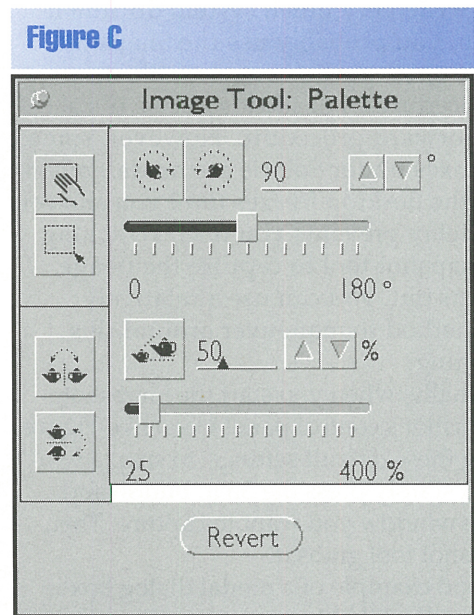
## Starting the Image Tool

The Image Tool provides some basic image editing features. To start the Image Tool, press the View button on the Snapshot window. Once you do so, you should see the Image Tool: Palette window, as shown in [Figure C](#). If you don't see this window, you can open it by selecting the Palette... option from the Edit menu.

From the palette, click the select area tool button . Now select the dialog box you want, leaving as little extra room around the borders as possible. Then select the Save Selection As... option from the File menu to save your image. It doesn't matter what name you use to save it, since we'll be fine-tuning the image in just a moment.

## Fine-tuning your image

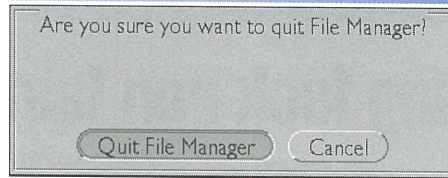
Once you save the selected area, the Image tool forgets the rest of your image. Now it's time to fine-tune your dialog box. To do so, first set the image size to 200%. You can do this by clicking on the text line to the left of the % sign and changing it to 200. Then click the resize button . With the image magnified, select the part of the dialog box you want to concentrate on with as much precision as you can manage. Again, save the image. Then resize it to 50%, and save it under its final name. You should now have a clean image like the one shown in [Figure D](#).



The Image Tool's palette offers you some tools you can use to edit your image.

We selected the 200% and 50% settings, not because they make the image as large as possible, but because these settings give the best results. Doubling the size of an image requires only that the Image Tool repeat each pixel. Shrinking by 50% lets the Image Tool reverse the process. If you magnify by any other size, the Image Tool will introduce rounding errors into your image, possibly degrading it when you shrink it the final time. ❖

**Figure D**



Once you complete the procedure, you should have a screen shot suitable for inclusion in your documents.

## WEB SURFING

# Netscape Navigator available for Solaris Sparc and x86

Surfin' the 'Net has caught on—and in a big way. Some of us use the Web for shopping, some for customer support, and some just for entertainment. Everyone's trying to cash in on the craze and catch your attention. As a result, Web page designers are using every possible feature to add some flash and dash to their Web pages. In this environment, it comes as no surprise that many Web pages are customized to work best with a specific Web browser. It's common to see Web pages that tell you "Best when viewed with browser X."

If you want to see these Web sites in all their Java-driven and frame-ridden glory, you'll need the latest and greatest Web browsers. One of the most popular Web browsers today is Netscape's Navigator, as evidenced by the number of Web pages containing the message "Best when viewed with Navigator 2.0." For this reason, you'll be pleased to know that you can get Netscape for Solaris on both Sparc and x86 platforms.

## Netscape Navigator v2.01

The current shipping version of Netscape Navigator for Solaris, version 2.01, supports many of the latest features, such as Java applets, tables, frames, and other new HTML features. And Netscape is already at work adding new features for Navigator 3.0.

Netscape Navigator v2.01 is available for both Sparc and x86 platforms running Solaris v2.5 or v2.5.1. It runs under both

CDE and OpenWin. The software is in `pkgadd` format, so it's simple to install on your system.

## How do I get it?

If you need to support multiple users, you can get 10-user license packs, complete with manual and software on a CD-ROM, from SunSoft for \$295. If you want a single copy for your own use, you can get it from several companies. For example, EIS Computers, Inc., sells a single-user license of Navigator 2.01 on CD-ROM for \$42. (You can visit their Web page at [www.eis.com](http://www.eis.com), or you can call them at (800) 351-4608.)

## The future...

If you don't mind being on the bleeding edge, you can download the latest beta version from one of Netscape's FTP sites. If you do so, you'll save a few bucks—and get a browser with even more features than Netscape 2.01. At the time of this writing, version 3.0 is in its fifth beta release.

You need to be aware of several shortcomings. If you plan to work with a beta release, Netscape puts a timeout on its beta releases, so you'll be able to run them only for a month or so before having to find a new beta or purchase a commercial version. In addition, beta software isn't known for its stability—it could crash at any time. Currently, Netscape doesn't put the software in `pkgadd` format, so it's a bit harder to install. ❖

Check out the additional info on page 2!

## Making fsck run faster on bootup

When you boot Solaris on your computer, it goes through many steps to prepare the environment for you. One of the many things it does is check the file system for errors. If you have many file systems on multiple disk drives, this process can take quite a while. In this article, we'll show you a couple of ways to make this process take as little time as possible.

### An introduction to */etc/vfstab*

As you know, Solaris uses many text files to configure the system. As far as running `fsck` during startup is concerned, the only important file is */etc/vfstab*. This file describes the layout of your file system. Figure A shows the */etc/vfstab* file for a small Solaris machine.

### The fsck process

When you boot Solaris, it will scan the */etc/vfstab* file and run `fsck` on all the file systems found. The column to watch for is the `fsck pass` column. Before the booting process completes, Solaris checks all the file systems that have a nonzero number in the `fsck pass` column.

The `fsck pass` column tells Solaris in which order to check the file systems. If multiple file systems have the same `fsck pass` number, then Solaris checks them simultaneously.

There's one basic rule you should use to get the most performance from `fsck`: Make sure that `fsck` checks only one file system per physical hard drive at a time. When a hard drive reads data, sometimes it has to move the heads. Each time you move the heads on a hard drive, you incur a time penalty. The farther you have to move them, the more time you must wait.

Each file system is constrained to a group of cylinders. When `fsck` checks a file system, the heads will move to the set of cylinders holding the file system. If you `fsck` only one file system, then the heads usually won't have to move to read the next set of data, so it can read that disk very fast.

However, if `fsck` is checking two different file systems on a hard drive at the same time, the heads will alternate between file system cylinder groups like a Ping-Pong ball. All this head movement will slow down the `fsck` process for both file systems. That's why you want to check as many different file systems as possible, but only one per physical hard drive.

Please note that when you run `fsck` on a group of file systems, `fsck` won't start checking any file systems in the next group until it finishes checking all of them in the current group. Thus, if you group large and small file systems, you won't get as great a benefit as you would if you group similarly sized file systems.

### RAM swapping penalty

The only possible snag with this technique is the effect that swapping has on performance. The more file systems you check simultaneously, the more RAM is required to do the job. If you check too many file systems simultaneously, then you can greatly increase the amount of swapping that Solaris must do. Obviously, swapping is a disk-intensive task, so you don't want `fsck` to check so many file systems at once that Solaris must spend too much time swapping.

Figure A

#device #to mount #	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
fd	-	/dev/fd	fd	-	no	-
/proc	-	/proc	proc	-	no	-
/dev/dsk/c0t0d0s1	-	-	swap	-	no	-
/dev/dsk/c0t0d0s0	/dev/rdisk/c0t0d0s0	/	ufs	1	no	-
swap	-	/tmp	tmpfs	-	yes	-

This */etc/vfstab* file describes the file system layout for a small Solaris machine.

Since `fsck` is a disk-intensive task, the last thing you want is a lot of unnecessary disk activity. The larger a file system is, the more memory `fsck` consumes when checking it. So, if you have a number of large file systems, you won't be able to check very many at a time before the system starts swapping.

Once the system starts swapping, `fsck` will operate more slowly. On some systems, the swapping overhead will more than offset the benefit of having `fsck` check multiple file systems at the same time. On other systems, the overhead won't be quite so bad. However, when you're checking a file system on a drive that also contains a swap partition, and the system is swapping heavily, then you'll still have the excessive disk head motion we discussed earlier.

When you're checking file systems on the same disk as a swapping partition, it will pay to be conservative: Check fewer file systems simultaneously to minimize swapping. Swapping also contributes to the other concern, SCSI channel capacity.

## Is your system swapping?

Once you come up with a plan where file systems can be checked simultaneously with `fsck`, you may want to see if checking them simultaneously will cause Solaris to swap. To do so, ensure that no one else is using your system and that you have no large processes going on, like OpenWin or CDE. Then, for each group of file systems you want to run simultaneously, run `fsck` on the file systems in the background. (You can do that by putting the `&` symbol on the end of the command line, so the `fsck` command will start and Solaris will immediately prompt you for the next command.) Then run `vmstat -S 10` to check the system's virtual memory statistics every 10 seconds, as shown in Figure B.

As you do so, watch the `sr` column. The larger the number in this column, the more swapping your system is doing. Before the system starts swapping, the `po` column will start growing. Once it's high enough, the system will start swapping and the number in the `sr` column should start increasing. As you can see, the value in the `po` column is large, but we haven't started swapping yet.

## SCSI channel capacity

If your SCSI channel is carrying as much traffic as it

can handle, you may not get a good performance gain from using this technique. Instead of waiting for disk head movements, `fsck` may wait for its data. When you have many simultaneous I/O operations, it's possible that the SCSI bus may become congested with I/O operations. This really isn't as much of a problem.

How many file systems can you check simultaneously without congesting the SCSI channel? It varies with the size of the file systems, the capacity of your SCSI channel, and the amount of RAM you have installed in your system. For a normal system, you probably shouldn't check more than four file systems at a time.

## Tuning your system

Now that you're familiar with the basic rules, all you need to do is to modify the `fsck` pass column in the `/etc/vfstab` file to make `fsck` check your file systems as quickly as possible. If you have a single hard drive on your system, there's not much you can do, except to be sure that you check only one file system at a time. However, if you have multiple hard disks containing multiple file systems, you should go ahead and start pairing file systems.

As you may expect, set the `fsck` pass column in `/etc/vfstab` to 1 for all the file systems that you want `fsck` to check first. Similarly, number all the file systems in the second group 2, in the third group 3, and so on. You can prevent `fsck` from checking a file system by using `-` (preferred) or `0`.

Although tuning a process that you can't get an exact measurement on is mostly guesswork, these guidelines will help you find a good starting point. If you have the time and the inclination, you can track the amount of time your system takes to boot and then make minor changes to see if you can tune it even further. ♦

Figure B

```
$ fsck /dev/rdisk/c0t0d2s4 &
$ fsck /dev/rdisk/c0t0d2s5 &
$ fsck /dev/rdisk/c0t0d2s6 &
$ vmstat -S 10
```

procs		memory				page				disk				faults		cpu					
r	b	w	swap	free	si	so	pi	po	fr	de	sr	f0	s0	--	--	in	sy	cs	us	sy	id
0	0	0	64328	840	0	0	1325	313	717	0	267	0	41	0	0	46	175	102	3	12	85
0	0	0	63524	504	0	0	2523	246	590	0	214	0	64	0	0	70	157	128	3	15	82
0	0	0	63524	652	0	0	2417	465	640	0	176	0	63	0	0	67	149	121	5	13	82
0	0	0	63524	456	0	0	1867	257	494	0	165	0	68	0	0	73	186	148	11	12	77

You can use `vmstat` to tell when the system is swapping.

SunSoft Technical Support  
(800) 786-7638

Please include account number from label with any correspondence.

## FREQUENTLY ASKED QUESTION

# Add swap space to your computer

**W**hen you run out of memory, it won't be convenient. You may be running a program that takes a tremendous amount of RAM, or you may be running a large mix of programs. In either case, you don't want Solaris to run out of memory. When you *do* run out of memory, you can add RAM or swap space.

While adding RAM will definitely make your computer faster, you have to buy it, turn the computer off, and install it. Adding swap space, however, is something you can do in just a couple of minutes, without even forcing your users to log off. In this article, we'll show you how to add swap space to your computer.

## Adding a swap partition

It's simple to add swap space to your machine if you have one or more unused slices on one of your hard drives. All you need to do is select the appropriate slice and tell Solaris that it's available for use as a swap area. You can do so with the `swap` command:

```
$ swap -a slicename
```

where *slicename* is the name of the slice in the *dev* directory. Please note that you have to be root to use the `swap` command. For example, to use `/dev/dsk/c0t2d0s4` for swapping, type:

```
$ swap -a /dev/dsk/c0t2d0s4
```

## Adding a swap file

You don't usually have spare slices on hard drives available for use. In this case, you may want to add a swap file to your system. To do so, you need to create a suitable file and tell Solaris to use it as a swap area.

You can't choose just any file to use as a swap file. Solaris has certain criteria it wants for a swap file. Luckily, they provide a program called `mkfile` that will create a file fulfilling these criteria. To use it, just type the command

```
$ mkfile -v size name
```

where *size* tells `mkfile` how much swap you want. You specify the size by using a number followed by a letter: k for kilobytes, m for megabytes, and b for blocks. The *name* field tells `mkfile` what name to call the file. We use the `-v` option to tell `mkfile` to tell us what's happening. Therefore, if you want to create a 20-megabyte swap file named *swap-area*, you could do so like this:

```
$ mkfile -v 20m swap-area
swap-area 20971520 bytes
```

Now that you've created your swap file, you can tell Solaris to use it. Again, you use the `swap` command. However, instead of specifying a slice of a hard drive, you use the absolute file name, i.e., the name of the file referenced from the root directory. Thus, if you created the *swap-area* file in */export*, you'd type:

```
$ swap -a /export/swap-area
```

## Performance tips

Before you add a swap area to your computer, think ahead. Here are a few tips that will let you get the best performance:

- Put your swap area on the drive that's used least.
- A single large swap area is more efficient than multiple swap areas on the same disk drive.
- Since Solaris alternates between swap areas, you can strategically place swap areas on specific drives to balance the load of disk I/O on your system.
- Be sure to add the swap areas to the */etc/vfstab* file to ensure that the swap areas will be available the next time you boot Solaris.

## Conclusion

Running programs that consume large quantities of memory doesn't have to force you to reorganize your computer. Adding swap space is a simple way to increase the system life of your computer. If you're careful, the result will have a minimal adverse effect on performance. ❖

