Solaris™ Solutions for System Administrators: Time-Saving Tips, Techniques, and Workarounds, Second Edition

Sandra Henry-Stocker
Evan R. Marks

Wiley Publishing, Inc.

Solaris™ Solutions for System Administrators

Time-Saving Tips, Techniques, and Workarounds, Second Edition

Sandra Henry-Stocker
Evan R. Marks

WILEY
Wiley Publishing, Inc.

Executive Publisher: Joe Wikert
Executive Editor: Robert M. Elliott
Assistant Developmental Editor: James H. Russell
Editorial Manager: Kathryn A. Malm
Assistant Managing Editor: Vincent Kunkemueller
Text Design & Composition: Wiley Composition Services

This book is printed on acid-free paper. ♾

Library of Congress Cataloging-in-Publication Data:

ISBN: 0-471-43115-X

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

*To the Boogie Man and the Possum for having the courage
to pull me from the recycle bin.*

— Sandra Henry-Stocker

*To my wife, Jodi, without whom I could never have
accomplished this task, and in memory of my sister Darci,
who taught me that positive thinking is a way of life.*

— Evan R. Marks

Contents

Preface

"The network is the computer." More than a catchy phrase, this slogan (which Sun coined more than a dozen years ago) describes a fundamental reality of computing today: The location of data, processing, and services are not as important as how and when these resources are used. With advances in the Web, we might even say, "The Internet is the computer." The systems we manage have evolved into portals to a universe of information and tools. Basic software applications are increasingly Internet-, intranet-, and Web-centric. This was true when this book was first published in the year 2000. It is even more true today with the proliferation of network storage and service-centric computing.

But how do we manage our systems and networks to best effect this reality? How do we keep our systems and networks manageable? What kind of foundation do we need to build locally so that the layers of complication wrought by services provided internally and externally go down smoothly and reliably? What kind of skills do we need to manage an increasingly virtual network? Our users are sending e-mail around the globe. Our networks comprise ever-changing configurations of new and aging systems. We are seeing both data and system services becoming disassociated from the server hardware. Our responsibilities as systems administrators now often incorporate managing clusters, configuring networks, monitoring security, Webmastering, performance analysis, and debugging, in addition to the standard fare of software installation, account and printer management, and backups.

Everything in this book is intended to move you in the direction of the manageable network—to introduce you to tools and provide you with strategies that will help you manage your systems.

No one can manage infinite complexity. And, if anyone could, that person would likely be smart enough or "interesting" enough that he or she wouldn't want to. Those of us who have been in the business of managing large networks for a long time understand that it is far easier to manage several hundred systems that are all the same than half a dozen that are all different. These two scenarios represent opposite ends of the spectrum with respect to homogeneity and heterogeneity. However, most of us are managing networks of systems that are somewhere in between these all-the-same and all-different models. Regardless of where you sit on the spectrum, you will find that your job will be easier and your life more pleasant in direct proportion to the degree of uniformity you can introduce into the environment that you manage. The more you can configure systems and services more or less identically, the easier a time you will have in every phase of managing them—from installation to backups.

At the same time, the move toward highly available services and virtual computing mandates a dramatic change in the way that we will do our jobs. Solaris 9 is more than just another release of Sun's operating environment. Solaris 9 represents a major shift of focus. Incorporating directory services and serious security tools (e.g., SSH, RBAC and the Sun Screen firewall) along with major new management tools, Solaris 9 will make your job much better or much worse, but clearly different. With enough insight and preparation, you will probably love this new release of Solaris.

Of all the strategies that we suggest to you within the pages of this book, the best is one that we've borrowed from the Boy Scouts: *Be prepared*. Regardless of how carefully and wisely you lay out your network and configure your systems and services, something will break. Adopting a policy of planning for disaster and of thinking through all the what-if scenarios that you can imagine will help reduce your stress as well as your downtime when one of these disasters actually happens. With respect to highly available services, preparing for disaster boils down to knowing how to manage and monitor HA services. Though we anticipate a steeply sloped learning curve for systems administrators coming up to speed on Solaris 9, we also have confidence that the plateau will be long and flat. Our networks will be easier to manage and our tasks less tedious.

Acknowledgments

Writing a book is always much more work than the authors imagine up front. There are reasons for this.

For one thing, even though most of us work between 40 and 60 hours a week and do our jobs well, it takes writing a book to make us realize how much we don't know. Sometimes, we only know eight ways to do something when there are 12 or more ways possible. Sometimes the numbers aren't that flattering. If ever you think you know it all, write a book. If ever you think you know nothing, train a replacement. We've come to understand that we're always teaching, and we're always learning. We learned a lot while writing this book.

For another thing, stuff happens. Each of us has a demanding job and a life. Each of us lost a beloved family member while writing the first edition of this book. Each of us fought off several types of microorganisms that preferentially attack people with kids. Each of us had times when our brains worked and times when they didn't.

Still, the book is done. There are also reasons for this. For one, we had help. We are so grateful for the people who filled in gaps in topics we only thought we fully understood, those who encouraged us and shared their insights, and those who cut us some slack simply because we were writing a book.

With respect to the first edition, Evan would like to thank Ken Baer, F. Steve Comins, and Chris Kordish at Sun for their sample scripts and topic ideas; Mark Henderson, James Redpath, and Ray Hildbrandt for their wonderful FAQs and insights into NIS+, the Starfire, and NVRAM; Evan Marcus, Jay Snyder, Don Lareau, and Deb Vitti for their impromptu reviews and topic recommendations; Jim Ford for his obsession with the English language and good grammar; and his wife and kids for their endless patience with this project.

With respect to the first edition, Sandra would like to thank the staff of ITworld.com (formerly Web Publishing) for their encouragement and their inspiring excellence; her niece, Rose Kenner, for helping her to be slightly more open-minded with regard to the Evil Empire; her daughters Danielle, Vail, and Nici for being proud of her (a singularly precious gift); and her husband, Eric, for doing all the cooking, cleaning, and laundry during the months of her book writing and repeatedly saying, over her objections, "It's okay. You're writing a book. That's important."

Both authors would like to thank Hal Stern and Ray Hildbrandt for their technical reviews and brilliant insights; Jeff Ruggeri and Bob Damato for jumping onto the bandwagon late in the process and adding a chapter (each) that we couldn't have written ourselves.

With respect to the second edition, Sandra would like to thank her coworkers at TCS for understanding when she was frazzled; Kevin Ho and Merle Ilgenfritz for coming to her rescue with material on Veritas Cluster and the Sun Fire line of servers when she was left to do this second edition by herself; and her wonderful husband for keeping her sane through yet another 6 months of no time for anything but work. She would also like to thank M-Systems for providing her with a 256MB Disk on Key (enabling her to carry all of the chapters and figures with her at all times); Bob Elliott and James Russell at Wiley for their support and sound advice during the preparation of this book, from proposal to printing; and James Russell a second time for his editorial advice; he played a surprisingly important role in shaping the book.

About the Authors

Sandra Henry-Stocker (aka S. Lee Henry) has been administering Unix systems for nearly 20 years. Her background includes 12 years as a columnist for SunExpert, Sun-World, and ITworld.com, and two terms on the now-defunct Sun User Group Board of Directors. She currently works for TeleCommunication Systems (a provider of wireless messaging applications) in Annapolis, Maryland, and teaches Internet classes at Anne Arundel Community College. She lives with her second family on a small farm on Maryland's Eastern Shore.

Evan R. Marks has over 10 years of experience in systems administration. He has published articles on high availability and has spoken at Sun conferences. Evan is currently Assistant Vice President of Infrastructure Engineering at MassMutual Financial Group in Springfield, Massachusetts.

About the Contributors

Bob Damato has many years of experience in network design (WAN and LAN), and Web and Unix administration.

Jeff Ruggeri is currently employed as a Solaris system administrator at MassMutual Financial Group in Springfield, Massachusetts. He has been hacking Unix in one form or another since he was about 13. His interests include randomness and searching for enlightenment.

Kevin Ho came to the United States in 1987 and received his Bachelor of Science degree in Electrical Engineering in 1990. He started his career as a Sun systems programmer in 1991 at the Computer Science Corporation where Sun systems were the predominant platform for software and firmware development. Kevin administered a hybrid network of Sun workstations, Sun servers, IRIX workstations, and IRIX servers. In 1998, Kevin received his Master of Science degree in Electrical Engineering in the subject of Communication Theory and Digital Signal Processing from the University of Maryland at College Park. He continues to work in the field of information technology and network engineering, and has accomplished many large-scale integration projects.

Merle Ilgenfritz is a security consultant and has been teaching computer technologies since 1992. He taught Microsoft-based classes for seven years for the corporate training departments of two colleges and a university in Rhode Island, where he resides. He is currently a certified Sun Microsystems instructor and course developer, teaching classes primarily at Sun's Burlington, Massachusetts, campus. He is certified to teach Sun's entire hardware product line, as well most of the storage and SysAdmin classes. He is married and has four children, four dogs, two birds, some rabbits, guinea pigs, chickens, fish, and who knows what else his wife has brought home today. . . .

Introduction

If you are familiar with systems administration but new to Solaris, we recommend that you read this book from cover to cover to give yourself a solid framework. Even if you're a seasoned Solaris sysadmin, you should still skim all the chapters, because we've crammed this book full of tricks and strategies that we've learned in our collective 36 years on the job that we think will help simplify and streamline your work. Many chapters in this book do no more than introduce a topic (such as LDAP). To do full justice to each of these topics would be to write a book twice the size of this one. Instead, we provide an introduction and encourage further reading.

Above all else, remember that there is no substitute for doing. To the extent possible, put what you learn to immediate use. This is the only way to fully understand any complex topic. Through the challenges that you encounter, you will earn your stripes and become confident of your skills and insights.

Conventions Used in This Book

The conventions that we use in this book are fairly simple. We use monofont for commands and **bold** when we tell you to type something or when we want to highlight something in a block of code for you to more easily notice it.

TIP Finally, keep an eye out for tips, like this one, as well as notes and warnings that we use to clue you in to real time-savers or other things that we want to call out to you.

What's in This Book

This book is organized as a series of chapters divided into four parts, which we describe briefly in the following sections.

NOTE Make sure to check out the book's companion Web site at `http://www.wiley.com/compbooks/henry`, where you can find links to the best Solaris resources on the Web and the latest versions of all the scripts mentioned in the book.

Part I: Setting Up Your Solaris Infrastructure

Part I covers basic network services as well as installation issues. Our goal in this part is to help you make decisions on the basis of your overall network so that your systems will be easy to manage—whether you have several systems or several thousand. This part covers everything from booting to backing up, with lots of attention paid to the tools—such as JumpStart and Web Start Flash—that can make your life so much easier.

Part II: Managing Your Systems

Part II takes you to the next step—what do you do when your systems are up and running. It covers managing and monitoring performance, volume management, and system security.

Part III: Looking After Your Hardware

Part III provides information on Sun hardware—from setting up and managing peripherals to configuring the awesome Starfire and Sun Fire servers.

Part IV: Surviving in the Real World

Part IV addresses critical issues in today's networks—managing the threats and opportunities presented by connecting to the Internet and coping with heterogeneity. We also share what we've learned over the years about managing our time so that we can be both awesome systems administrators and acceptable spouses and parents.

PART

I

Setting Up Your Solaris Infrastructure

No matter what anyone tells you (even us!), systems administration is as much an art as it is a science. Were you to systematically gather all the relevant facts before embarking on the task of setting up your systems, you would *still* encounter problems. Some systems would be overused, and some underused. Some disks would fill to the brim while others would be rarely used. Small projects would end up taking enormous amounts of time, while major ones would come to run themselves. The problem is simple—things change. The needs of users change, the software on the system changes, hardware breaks, upgrades arrive, people come and go, people change their minds, and estimates are only estimates. The best you can do is to configure your systems so that you can work in spite of the flux. And, yes, we *still* cling to the Boy Scout motto: Be Prepared!

Let us assume that the goal of systems administration is not to avoid problems, but rather to have processes and procedures in place that force them to approach you in single file (no pun intended). The manageable network is not one that runs flawlessly, but one in which there is time and capacity enough to resolve each problem without major disruption.

In this initial part of the book, we provide our thoughts and suggestions to help you accomplish this goal. We make suggestions about automating your system installation and patching by using JumpStart and Web Flash. We encourage you to plan file systems with reliability, security, performance, and manageability in mind. We provide insights into naming services such as DNS and Sun's NIS and NIS+—and we introduce LDAP, the naming service we see as in everyone's "cards." We detail the Solaris boot process as it moves from cold hardware to being fully operational. We describe Solaris run states and provide instructions on modifying the processes that start or shut down automatically. We discuss PROM-level diagnostics that you can use to isolate problems.

We don’t live in the world of infinite budgets and patient users, waiting for every problem to occur like the Maytag repairman. We assume that you don’t either. We’re not sitting in front of a pile of brand-new systems in an office space not yet filled with people and activity. We assume that you’re not either. Our networks are not composed entirely of Solaris servers and Solaris clients. We assume that yours are not either. We want leisure hours, happy families, and personal downtime. We assume that you do too. Our vantage point is that of system administrators with too much to do, working in an environment where change is the only thing that remains a constant. We’re sure you will find something in this part to make the setup and management of your systems less chaotic.

CHAPTER

1

Making Smart Decisions about File Systems

Laying out file systems on your servers and clients is one of the most important steps you will take in building a manageable network. How you arrange file systems on individual hosts and across a network can dramatically affect the performance of your systems and network, the reliability of your systems, security, cost, your backup strategy, and the amount of work that you have to do to maintain these file systems and the software and data they house. In fact, your file systems can easily be the single most significant factor in the overall performance and reliability of your network.

In this chapter, we will look at how file systems are organized and how this organization affects performance. We will also offer suggestions on how you might arrange file systems on a network to make your network more manageable.

Most of us learned to be system administrators in small steps. We covered the basics—file systems, permissions, creating user accounts—and then started learning more advanced skills as the needs and challenges presented themselves. Yet, even after as many years as we have been managing Solaris systems, we find that we're still learning. It goes with the job and the constant influx of new tools and technologies. Because none of us (unless you, dear reader, are an exception) knows all there is to know about Unix or, in particular, Solaris, this chapter attempts to answer the essential question that so many of us ask ourselves from time to time: What the hell is that? By describing the basic topology of Solaris and then detailing key system processes, configuration files, log files, and so on, we hope to fill in some of the gaps in your understanding of Solaris and provide you with a comfortable familiarity with many of the files and directories you'll run into.

File Systems from the Roots Up

With a single root directory (which is designated simply as "/") from which all branches of the treelike structure derive, the file system provides a degree of consistency across the most divergent variations of Unix. Yet the simplicity of this hierarchical file system (playfully depicted in Figure 1.1) hides an increasing degree of complexity with every major release of Solaris. Early versions of Unix systems employed a single disk-based file system. The current version of Solaris supports many different types of file systems—the customary disk-based fast file system; network-based file systems; compact-disc read-only memory (CD-ROM) file systems; redundant array of inexpensive disk (RAID) systems, in which many disks are combined into one logical file system; file systems on Disk Operating System (DOS) diskettes; and even pseudo or *virtual* file systems, which are composed not of files at all but of processes, network sockets, character device drivers, and the like. Even so, the convenient abstraction of the familiar treelike structure persists, and most Unix users are oblivious to the complex structures upon which this abstraction is built.

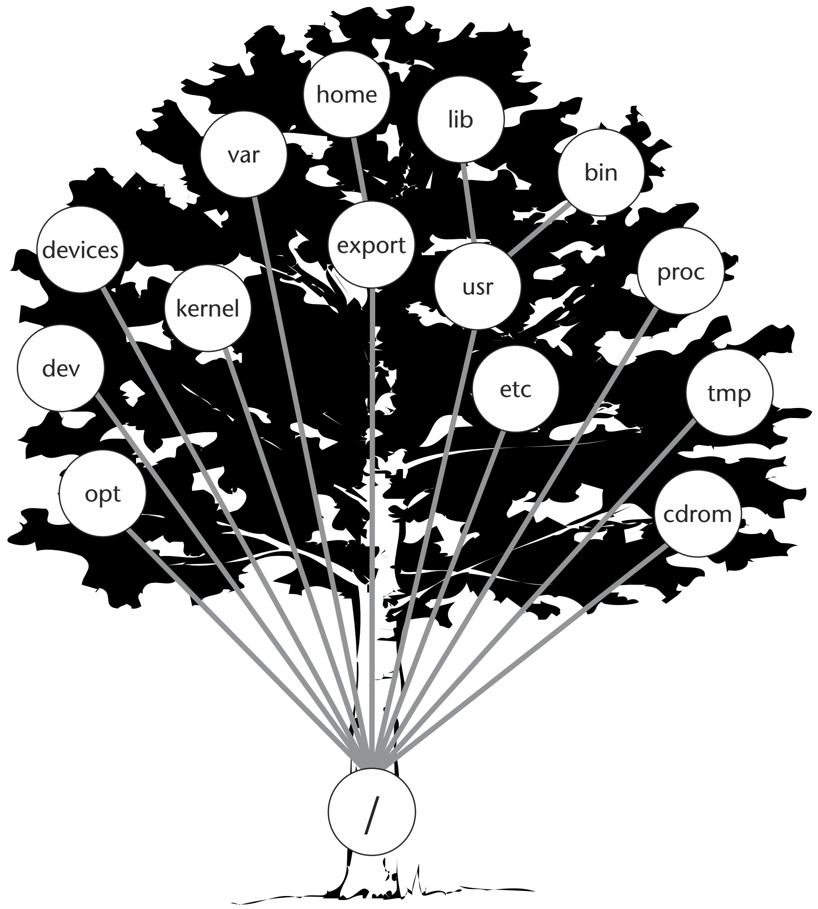


Figure 1.1 The Solaris file system tree.

To begin our examination of Solaris file systems and how you can optimize the performance and utility of file systems across your network, let's first examine a traditional Unix file system (UFS) from the ground up. A single file is, first and foremost, a collection of data—whether ASCII or binary—that is stored in contiguous and/or noncontiguous chunks on a disk. These chunks are sized according to a storage unit called a *block* (for UFS, the default block size is 8,192 bytes). As you may already have noticed, directory files are always some multiple of 512 bytes. When you first create a directory with `mkdir`, you will notice that it looks something like this:

```
solaris% mkdir mydir; ls -ld mydir
drwxr-xr-x   2 sdraven      staff         512 Nov 11 11:15 mydir
```

and, later, as you add files, it jumps in size:

```
solaris% mkdir mydir; ls -ld mydir
drwxr-xr-x   2 sdraven      staff        1024 Nov 22 16:05 mydir
```

This is because space for directory growth is allocated in 512-byte units. By preallocating space in directories, directories can grow without as much overhead as if space had to be added each time the contents of a directory changed.

Files, of course, don't grow in units of blocks, but by the amount of data they contain. The space allocated for them, however, does grow in units of blocks, allowing them to grow up to the next increment of a block before additional disk space must be allocated. In general, all but the last block of a file is full. Figure 1.2 illustrates disk space allocation for a regular file.

Some file systems milk this efficiency even further by allocating more than a single additional block when a file grows. These are referred to as *extent-based,* as opposed to *block-based,* file systems. Allocating a larger amount of disk space allows the files to grow more contiguously. On the other hand, an extent-based file system runs a greater risk of wasting space that is allocated and not used.

The separate blocks of data that make up a single file are linked to each other by on-disk record structures, which store the location of each of the blocks. This structure is sometimes referred to as a *block map.* In a similar manner, the available disk space in a file system is maintained in a free-block list. A more complex record structure is used to store the descriptive data concerning each file, sometimes referred to as *metadata* (a common term for data that describes data). This metadata includes the file's owner (i.e., the user ID [UID]); the file's associated group (the group ID [GID]); the permissions matrix; the file creation, modification and access times; the size and type of the file; and so on. In fact, the only items not stored in this structure are the contents of the file (as mentioned, this is stored on disk) and the file's name and location within the file system.

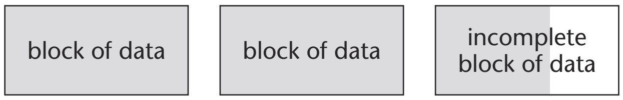


Figure 1.2 Blocks of data making up a single file.

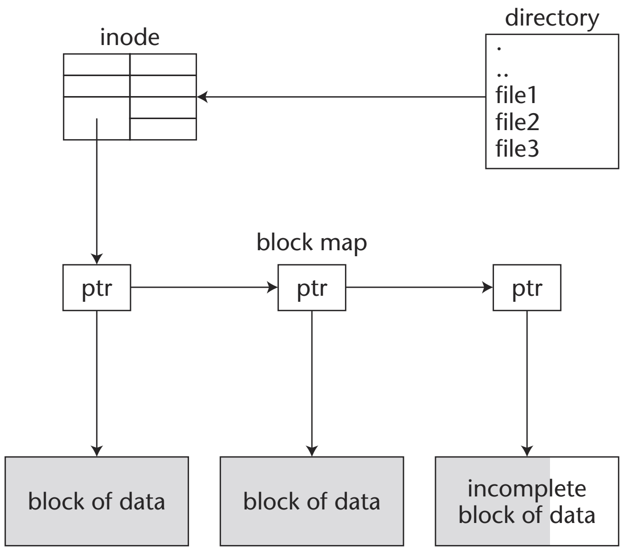


Figure 1.3 File structures.

A file's name is stored within another file called the *directory file.* The inclusion of the filename within a particular directory also determines where it "lives" within the tree structure. This location has almost nothing to do with the location of the data itself. In fact, the same file can exist in any number of locations, even in the same directory more than once if it has more than one name. If a single collection of data blocks constituting a single file has more than one file system identity, this duplication is only expressed through the separate directory entries and in the number of links indicated within the inode. As you might suspect, directory files also contain the inode numbers of the files they contain. This provides a way for the file system to easily retrieve the metadata when a long file listing is requested with the `ls -l` command. Figure 1.3 illustrates the components of a single file in a traditional Unix file system. These include the directory file that provides a file system entry, the inode that contains the metadata (owner, permissions, etc.), and the pointer (`ptr`) records that link the records in the block map and point to the file's data stored on disk.

File Systems: Shortcuts to Data

A file system can be said to be an interface to files—a way to address files and access their contents. When a user issues an `ls` command, for example, the file system receives a request for the contents of a directory file. If the user has execute permissions, the contents of the directory will be displayed. If the user issues an `ls -l` command, additional resources must be tapped. Information stored in the associated inode must also be read and displayed. In addition, some of this information must first be resolved; the UID and GID fields will be looked up in the password and group files (or maps), and the textual name will be displayed in place of the numeric identifiers.

Adding a single line to a file can cause it to extend beyond its current last block and require allocation of another block. The new block must be added to the block map. The length of the file, as noted in the inode, must be changed and the modification time updated to reflect the change, as well.

If a hard link is created to an existing file, the inode field containing the number of links must be incremented. If a hard link is removed, the field must be decremented. If the number of links field drops to 0 (i.e., all instances of the file have been removed), the file space can then be reclaimed and the blocks can be restored to the free list. Figure 1.4 displays the file structures associated with two hard-linked files. As you can see from this figure, the only additional resource used by a hard link is the extra directory reference. The other file structures are shared with the original file.

Adding or deleting a file involves a similar sequence of operations. When a file is added to a directory, an inode is reserved, space is allocated, the directory file is updated with the name and inode number of the new file, and the block map is updated. When a file is removed, its blocks are added to the free-block list, the directory file is updated as the file's entry is removed, and the inode is made available (provided there are no other links to the file).

When a user reads a file, the directory identifies the inode, which is used to determine file access privileges and locate the file on disk. In fact, a file's access permissions are used in any file operation to determine if the individual performing the operation has the proper permissions to do so. The file's content is then retrieved from disk.

If a file is moved, one or more directory files must be updated. If it is moved within a directory, a single directory file is modified. If it is moved from one directory to another, two directory files must be modified. Neither the block map nor the inode associated with the file is affected. If a file is moved from one file system to another, however, the process of establishing it on the new file system is much like that of creating a new file. The existing inode can no longer be used because it belongs to the initial file system.

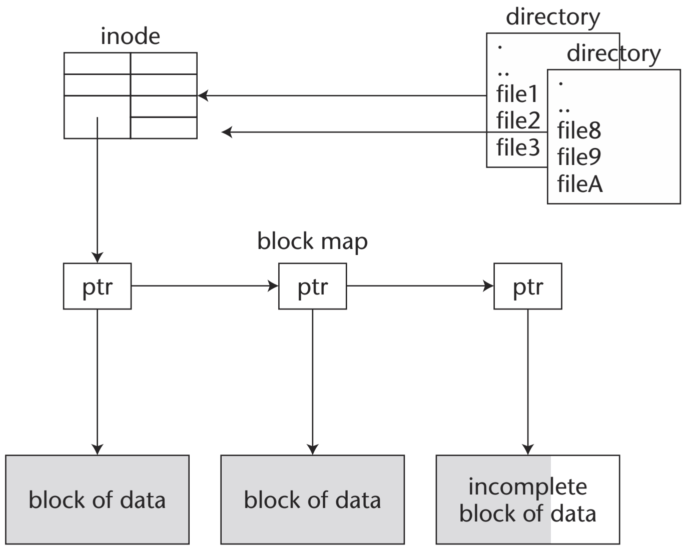


Figure 1.4 Hard links and associated file structures.

Types of File Systems

There are a number of file systems that Solaris can support, whether out of the box or with some additional software. Before we get into all of the file systems, we will briefly describe the major categories of file systems: disk-based, memory-based, network-based, and so on. The framework within which these different types of file systems are supported simultaneously has come to be known as the *virtual file system.* The generalization of the inode structure into the vnode makes allowances for the non-UFS types.

The basic difference between different disk-based file systems is how they store and access data. Earlier in this chapter, we described how UFS structures such as inodes, block maps, and directories are used to organize data and make it available. It's easy to imagine a file system without directory files; it would simply have a "flat" file space, though it might still use inodes and lists to keep track of files as well as free and used space. UFS, however, is hierarchical, as are the vast majority of file systems that are available today. It may help you to picture each file system as a logical structure between you and the data on disk that allows you to find and use portions of that data that you think of as files.

Regular (disk-based) file systems include the types shown in Table 1.1.

UFS is the standard Berkeley fat fast file system (FFFS) and the one we have been discussing. The file system is called "fat" because, unlike its predecessor, it allows for certain structures, such as the number of inodes per cylinder group, to be established when a file system is created. UFS is the file system you will most commonly encounter on Solaris systems. Other file systems listed in Table 1.2 are third-party file systems with features similar to those of UFS.

TIP VxFS from VERITAS and QFS from LSC provide some advantages over UFS, primarily by providing for extremely large files and file systems. For UFS, the maximum for files and file systems is a terabyte; for the others the maximum is, measured in petabytes (8 PB or 8,000 TB).

Table 1.1 Regular File Systems

FILE SYSTEM TYPE	MEDIUM	DESCRIPTION
UFS	Disk-based	The Berkeley fat fast file system; Solaris default
VxFS	Disk-based	VERITAS file system
QFS	Disk-based	LSC Inc.'s file system
pcfs	Disk-based	MS-DOS FAT file system; floppies
hsfs	Disk-based	High Sierra file system; CD-ROM
udfs	Disk-based	Universal Disk Format; DVDs
tmpfs	Memory/disk-based	Temporary file system; swapping

Table 1.2 File System Comparison

	UFS	VXFS	QFS
Maximum file size	1TB	8,000 TB	1PB
Maximum file system size	1TB	8,000 TB	1PB
Allocation method	Block-based	Extent-based	Extent-based
ACL support	Yes	Yes	No
HSM support	No	Yes	Yes
Logging (type)	Yes (metadata)	Yes (metadata/data)	No
Direct I/O	Yes (Solaris 2.6 and later)	Yes	Yes
Quotas	Yes	Yes	No
Page cache	Yes	No	No

Another difference between UFS and these high-capacity file systems is whether the file system is block- or extent-based; the difference here is whether files are extended a block at a time or a number of blocks at a time. A third difference is whether there is support for hierarchical storage management (HSM), in which data transparently flows from faster to slower media when it has not been recently accessed; HSM is not available in UFS, but is available in VxFS and QFS. Some of the major differences between these and the UFS file systems are shown in Table 1.2.

If you need file systems larger than 1 TB or you require some of the other features shown in Table 1.2, the article by McDougall cited in the Bibliography and Recommended Reading provides further insights into the trade-offs of using these products instead of UFS.

In addition to Unix file systems, Solaris supports pcfs- and hsfs-type file systems. These types of file systems are most commonly encountered when you deal with diskettes and CD-ROMs, but are not necessarily restricted to these media types. Solaris 8 and later also support UDFS (the Universal Disk Format file system) used for DVD drives.

The *pcfs*-type file system implements the MS-DOS FAT file system. Once mounted, standard Unix commands such as `ls` and `cd` can be used with these files. One should be careful, however, in managing the differing line termination conventions found in MS-DOS and Unix files. Whereas MS-DOS files end lines with both a carriage return and a line feed, Unix files end lines with only a line feed. The `unix2dos` and `dos2unix` commands overcome this minor inconsistency quite handily.

The *hsfs*-type file system is the High Sierra file system, also referred to by the name *ISO 9660.* The Solaris implementation of hsfs also supports the Rock Ridge extensions that provide for Unix-style file naming. While mounted, standard Unix commands can be used to access files and move around in directories. However, the underlying geometry and structure of an hsfs-type file system is dramatically different from that of UFS—and, of course, you cannot write to files on a CD-ROM.

File systems of type *tmpfs* may look and act like disk-based file systems, but they actually reside in memory more than on disk. The primary use of /tmp is for memory swapping. Data is not written to disk unless physical memory is exhausted. The contents of tmpfs file systems are, therefore, temporary. In fact, the /tmp file system, the only file system of this type on most Solaris systems, is emptied on a system reboot. We briefly describe swapping a little later in this chapter.

The *Network File System* (NFS), now available on virtually every operating system, ties systems together by providing an intuitive file-sharing mechanism across a network and divergent platforms. Happily, differences in the underlying system's processing (e.g., whether the processor orders bytes into words using the big-endian or little-endian method) are transparent. With automount support, NFS performance can be greatly improved. When file systems are mounted as needed and unmounted after a period of nonuse, considerable file system overhead is avoided.

Pseudo-File-Systems

Although pseudo-file-systems may appear as file systems to users, they are actually abstractions through which various system and process data can be accessed. *Pseudo-file-systems* are used to represent such things as processes, network sockets, device drivers, and first–in–first–out structures (FIFOs). In our discussion of the /proc file system, we will clearly show the benefits of accessing information about running processes through pseudo-file-system abstraction.

Pseudo-file-systems include those listed in Table 1.3.

Table 1.3 Pseudo-File-Systems

FILE SYSTEM	DESCRIPTION
NFS	Network File System (generally remote UFS)
cachefs	Local disk-based cache for NFS file systems
autofs	Used for automatic mounting via NFS
specfs	Access for device drivers in /dev
procfs	Access to running processes and kernel structures
sockfs	Access to network sockets
fifofs	Access to FIFO structures
lofs	Loopback file system; used to create alternate paths to existing files

File Types

The classification of files in Solaris can be confusing to anyone not used to it. The first breakdown is that of regular versus special files. The classification of *regular files* encompasses a wide range of file types as users think about them—binaries, scripts, data files, configuration files, and so on. The organizing element is this: The kernel doesn't make any distinction between any of these file types. Differences between them exist only at the user level (e.g., the content, whether they are executable, and so on). File types that don't fall into the regular category are links, pipes, sockets, and so on. These files are recognized as being different and are treated differently by the kernel. Table 1.4 lists file types and the characters used to designate them in a long listing (i.e., `ls -l`).

Although the Solaris kernel does not differentiate between different types of regular files, users do. So do windowing systems. For this purpose, there is an underlying class structure that identifies files by type. This structure enables the expected results to happen when a user double-clicks on an icon within the file manager tool or drops it into another window (e.g., a print tool). In addition, the `/etc/magic` file is used to identify file types using embedded magic numbers. Not all file types have magic numbers, of course. For those that do, the offset (generally 0), type (length), and the identifying pattern are specified in the `/etc/magic` file. The entry `0 string %PDF-1.2 Adobe Portable Document Format (PDF)` specifies that version 1.2 of the PDF format is identified by virtue of the fact that its files begin with the string `%PDF-1.2`. A user can determine the file type of a specific file by issuing the `file` command. This command looks at the first several bytes of the file and references the `/etc/magic` file to determine the file type.

Table 1.4 File Type Designators

FILE TYPE	DESIGNATOR
Regular file	-
Directory file	d
Block special device	b
Character special device	c
Name pipe (FIFO)	p
Symbolic link	l
Door (Solaris 2.6 and later)	D
Socket	s

```
file *
subdir:             directory
dumpxfer:           executable /opt/LWperl/bin/perl script
log.out:            ascii text
logmon:             executable /bin/ksh script
mailman:            executable /bin/ksh script
processlog:         executable shell script
watcher:            executable /bin/ksh script
nightlyreport:      executable c-shell script
killit: whois:      ELF 32-bit MSB executable SPARC Version
1, dynamically linked,stripped
```

You can add file types to `/etc/magic` by editing the `/etc/magic` file and placing the offset, string, and type in the file. For example, if you have a type of file called a *testfile,* which always starts with the string `test`, the following `/etc/magic` entry would be coded:

```
#offset     type   value  File Type
0           string test   testfile
```

With this entry, if you have a file called `myfile` with contents `testing 1 2 3`, the file command would identify the file type as follows:

```
# file myfile
myfile:   testfile
```

Taking advantage of `/etc/magic` is a great way to help your users identify their files. Consider defining file types for any major applications that your users access.

If you have never browsed through the wide range of file bindings available in your windowing system's binder tool, you will probably be impressed by the variety of file types that the windowing system can recognize and treat differently, although you will probably never see most of them. The kernel knows only the eight types listed in Table 1.1. Figure 1.5 illustrates a breakdown of files by type from a user's point of view. The file types recognized by the kernel are circled. You can see from this incomplete breakdown that differentiating files by type can get to be quite arbitrary. The important thing to remember is that, as far as the kernel is concerned, a regular file is a regular file. It will not try to stop you from making a file of C source code executable and running it any more than it would try to prevent you from printing a binary; you'll simply get some very odd results.

Directory files, as we've mentioned earlier, are special files that contain the names and inode numbers of the files they "contain." We put that word in quotes for a good reason: the relationship between directories and files exists only because of these entries. Directory entries also contain a file name length for each entry.

NOTE By the way, you might want to look at the include file `dirent.h` if you are curious about the format and contents of directory files.

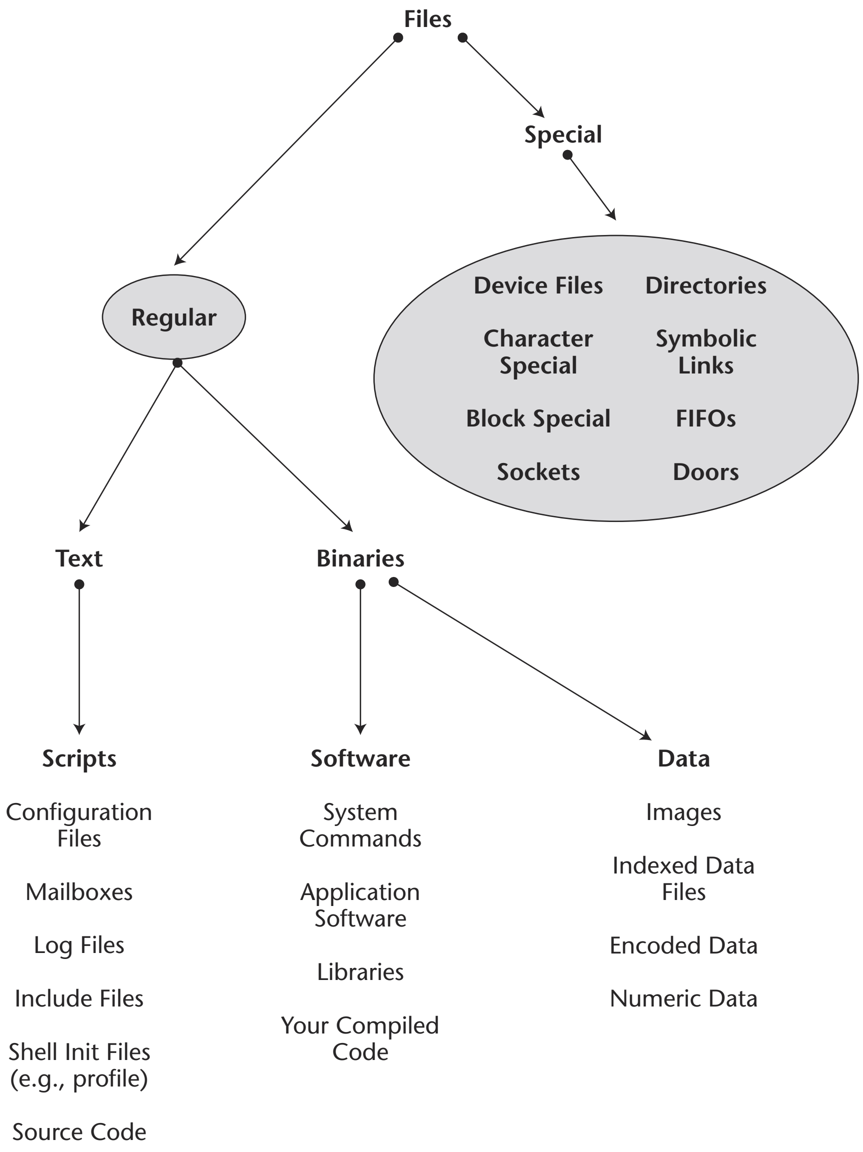


Figure 1.5 File types.

Block special device and *character special device files* are unusual in that they do not contain data. Meant solely as access points for physical devices such as disks and tapes, these files provide the means to access the related device drivers. The main difference between the two is how they handle input/output (I/O). Block-special device files operate on blocks, while character-special device drivers work on a character-by-character basis.

Sockets are special files used for communications between processes, generally between systems. Client/server applications, as well as many network services, use sockets.

Named pipes—also called *FIFOs* for "first in, first out"—are special files used for communications between processes, generally on the same system.

Symbolic links are pointers to other files. They are similar but not identical to shortcuts in Windows. Symbolic links—sometimes called *symlinks* or *soft links*—contain the names of the files they point to.

Hard links, as you may already know, are not special files at all. As mentioned earlier in this chapter, a hard link is indistinguishable from the file it links to—this is not true with symbolic links—except for their locations within the file system. In fact, either could be called the link or the file. There is no difference.

The Solaris File System

The layout of Unix file systems has maintained a certain consistency from its early days to now. Under the surface, however, a number of new file system types and directories have evolved. These include, but are not limited to:

/etc/rc?.d and **/etc/init.d** Used in booting and in changing run states.

/proc An interface into running processes.

/tmp Primarily used as swap space and no longer a standard Unix file system.

Directories

There are many special directories within each of these file systems that stand out as being worthy of individual explanation and attention. These are the directories in which major system functions are configured and/or managed.

/etc The host-specific configuration files are stored in `/etc`. If you are not running NIS or NIS+, the `hosts`, `passwd`, and `shadow` files determine which hosts your system knows about (aside from DNS) and who can log in. Other files include important configuration files—for example, those for the `init` (`inittab`), the `inet` daemon (`inetd.conf`), and the `syslog` facility (`syslog.conf`). The `/etc` directory also includes the configuration file for `sendmail` (`/etc/mail/sendmail.cf`), along with other `sendmail` files, and, at many sites, also contains configuration files for Web and news servers. We discuss some of the directories here.

/etc/cron.d Configuration information for `cron` (but not the `cron` files themselves) are stored in this directory. The `cron` files for each user are stored in `/var/spool/cron/crontabs`. The files for the `at` command are in `/var/spool/cron/atjobs`.

/etc/default The default settings for many services (e.g., policies about whether root can login remotely) are stored in this directory. We discuss the contents of `/etc/default` later in this chapter.

/etc/dfs This directory contains the `dfstab` file, which determines what file systems are shared and how.

/etc/mail This directory contains the configuration files for `sendmail`.

/etc/init.d This directory is the primary location for start/stop scripts. All start/stop scripts should be located here and in whichever of the `/etc/rc?.d` directories correspond to the run states in which they should start or stop.

/etc/rc?.d These are the run-state-specific directories for start/stop scripts (`/etc/rc0.d` is for run state 0, `/etc/rc1.d` is for single-user, and so on). Any files starting with the letter *K* are executed by the corresponding `rc` script (e.g., `/usr/sbin/rc2`) when it enters the particular run state. Afterward, any files starting with the letter *S* are executed. These scripts are executed in alphanumeric order (e.g., a file named *S88httpd* would run before one named *S99mydaemon*).

/dev and **/devices** These directories are the primary location for device files. The most commonly used tape device, for example, is probably `/dev/rmt/0`, and the most common root directory is probably `/dev/dsk/c0t3d0s0`. For tape devices, the letters in the name indicate how the device will operate—for example, whether the device will use compression or fail to rewind after use. For disk partitions, the characters in the name specify the controller number, target, disk number, and slice.

/usr/bin The bulk of the system commands are located in the `/usr/bin` directory. The `ls` and `echo` commands, for example, are located here. Some commands (those with origins in the Berkeley side of Solaris's ancestry) are located, instead, in `/usr/ucb`. There is also a `/usr/sbin` directory and a `/usr/local/bin` directory on many systems.

/usr/man The system manual pages are stored in this directory. However, the directory is usually a pointer to `/usr/share/man`. Most sites that supplement their systems with tools and applications, including their own man pages, store these man pages in `/usr/local/man`; for the most part, this is a better choice than individual man directories within the application directories because it's easier to set up users' MANPATH variables.

/usr/local As previously mentioned, this is a site-specific application directory, often maintained on a file server. Contrast this with `/opt`.

/usr/local/bin This is a site-specific directory of tools and utilities (*bin* stands for *binary*). It is also a good place for you to put scripts that you want your users or fellow sysadmins to be able to run.

/usr/local/man This directory is often used for man pages associated with add-on applications.

/usr/include This directory stores the header files for programming and descriptions of system structures (e.g., the record structure for an inode). Subdirectories contain header files for specific functions (e.g., networking or file systems). A system that is not set up to provide developer support will generally not have this directory.

/usr/lib/acct This directory holds accounting commands.

/usr/platform This directory contains platform-specific files—for example, the files to create a bootblock for a specific system type.

/var/log Most system log files are stored somewhere in the `/var` directory. The syslog logs (i.e., `syslog` and `syslog.?`) are stored in `/var/log`. The messages file and login files are stored in `/var/adm`.

/var/adm This directory contains other log files (e.g., `messages` and `wtmp`). The fact that these files are here rather than in `/var/log` does *not* imply that they are maintained by something other than the `syslog` facility. The `messages` file is a syslog-maintained log file. To see what files are maintained by `syslog` and which are not, look at `syslog`'s configuration file—`/etc/syslog.conf`.

/var/mail This is the directory where user e-mail is stored. It will be populated with files that have the same names as usernames on your system (e.g., `/var/mail/mallory`). You may also see some other files in this directory from time to time. Lock files (e.g., `/var/mail/mallory.lock`) and popper files (e.g., `pop.mallory`) will appear while users are accessing their mail. You may have to remove these files manually when a mail client crashes or is otherwise stopped improperly before the user can again access mail.

/var/yp This directory houses the configuration files for the NIS service on a server. You should see a Makefile; this file contains instructions of what `make` will do when you update an NIS map (e.g., `make passwd`). The times that maps are updated are captured in files with names such as `passwd.time`. The actual data files are kept in a subdirectory named after your domain. For example, the directory `/var/yp/foo.com` would house the data files for Foo's NIS domain. These files are named `passwd.dir`, `passwd.pag`, and so on.

/var/tmp This is temporary space for users (preferably used in place of `/tmp`). Applications requiring temporary space are usually configured to use this location as well. If you look into the `/var/tmp` directory, you may see files of many types. Some of these (e.g., those with names such as `Ex0005757577`) are temporary files used by an editor such as vi.

NOTE **The `/var/tmp` directory is *not* used for swapping, but provides temporary storage that anyone on the system can use. Before Solaris 2, `/tmp` was used for this purpose.**

/var/sadm The `/var/sadm` directory contains information about packages installed on the system. The `/var/sadm/contents` file, for example, contains information about every file loaded during installation, along with information detailing its size and permissions when it was installed. This file is used with the `ASET` (security auditing) tool to detect files that have changed since installation. Obviously, some files need to change (e.g., your `/etc/hosts` and `/etc/passwd` files). You should not be alarmed if ASET informs you that they have changed. This file is also used by the `pkgrm` utility so that it can locate all of the files that it must remove when uninstalling software for you.

UFS Logging

Beginning with Solaris 7, the standard Solaris UFS file system has incorporated a logging function that, while transparent to users, makes the file system considerably more reliable. Those of us who remember sitting through painfully long `fsck`s will probably be delighted by the "File system is stable" messages that we now usually see when rebooting a system that has crashed. The reason for the change is logging ability. Because UFS file systems now maintain a log detailing updates to files, they have the means for recovering the state of a file system without an extensive analysis and rebuilding by `fsck`.

`fsck` makes five passes over a file system, checking different file system structures in each pass, as follows:

1. Blocks and sizes
2. Pathnames
3. Connectivity
4. Reference counts
5. Cylinder groups

By examining the file system from each of these perspectives, `fsck` is able to repair damages resulting in a file system that is once again consistent.

Most file system inconsistencies in the past were due to the constant state of fluctuation that file systems are in and the fact that changes to file systems usually require several operations, as shown in the previous list. Modified data blocks must be written back to disk; additional blocks must be allocated and represented in the block map; inodes must be created, removed, and updated; and directory structures must be updated to reflect new files and files that have been deleted or moved. If a crash occurs when some, but not all, of these changes have taken place, an inconsistency is created. The `fsck` tool attempts to resolve this kind of inconsistency by examining the various structures and determining how to complete the interrupted file system changes. It examines inodes, the block map, and directory files in an effort to piece together a consistent file system by adding, removing, or modifying file system structures. For example, if a directory contains the name and inode number of a file that no longer exists, `fsck` can remove the reference. When `fsck` knows nothing about the state of a file system that it is asked to check, it is extremely thorough and time-consuming in examining and repairing it.

The way that logging works in Solaris today is as follows: There is a logging file structure in the same partition as the file system. Before any change is made to the file system, a record is written to this log that documents the intended change. The file system change is then made. Afterward, the log is once again modified—this time to reflect the completion of the intended change. In other words, the log file maintains enough information about the status of the file system to determine whether it is intact. If there are outstanding changes at mount time, the file structures associated with those changes are checked and adjusted as needed, and an examination of the entire file system is avoided.

Working with File Systems

Managing disks and the data that they house is one of the primary responsibilities of system administrators. After all, nothing is as fundamental to a computer system as the data and processes that make up the computing environment. In this section, we look at file system and the disks on which they are built and offer some insights on how to work with yours.

Building and Maintaining File Systems

With respect to function, there is little that is more fundamental to system maintenance than your users' access to their files. Maintenance of file systems—ensuring their availability and integrity—is, therefore, one of the most important aspects of any sysadmin's job.

For many sysadmins, the task of building file systems is simply something that happens during installation. Whether you use the automatic layout feature or profiles in JumpStart, the building of file systems is automated. If you are adding a new disk to a system, on the other hand, you may need to repartition it and manually construct file systems to your own specifications. Once the disk has been physically attached to the system, you want to be sure that the system recognizes it. For Small Computer System Interface (SCSI) disks, be sure that the SCSI target is unique for the intended bus. On a system that already has a `/dev/dsk/c0t3d0` and a `/dev/dsk/c0t0d0` disk, the most common next choice is `/dev/dsk/c0t1d0`—SCSI target 1.

WARNING We recommend attaching new drives when the system is halted to ensure that no stray signals are communicated to the circuitry during the insertion. Some disks are designed for "hot swapping" (insertion while the system is running). All others should be removed or attached after the system has been powered down.

When you first power up a system after adding a drive, you should stop at the boot monitor (i.e., hit L1.A or Stop-A before it gets too far along in the boot process) and type **probe-scsi** at the `ok` prompt. With OpenBoot 3.0, the recommended procedure is to use the command `setenv autoboot? false`, and then do a reset. This interaction should look something like this:

```
Type 'go' to resume.
Type help for more information.
ok probe-scsi
...
Target 1
  Unit 0  Disk Seagate   ST1523ON  1234
```

The preceding information shows you that the new disk is recognized. If you don't see your disk at this point, you probably have a problem with your SCSI cabling or with the disk itself. It's pointless to proceed without resolving this problem.

In order to build support for a new disk, you need to reboot with the command **boot -r**. This command (r stands for “rebuild”) instructs the system to configure the device support needed for the new disk. If for some reason you cannot easily boot the system at the console, you can effect this command indirectly by touching the file `/reconfigure` and then rebooting the system. (Yes, you can do this remotely.) The boot sequence will then include the rebuild and remove the file, preventing the rebuild from occurring during every subsequent reboot. The correct device nodes for the disk are automatically added. If you cannot reboot, or if you have rebooted but forgot the *-r,* you can simulate the reconfiguring boot with the following set of commands:

`drvconfig`. Reconfigures the drivers

`disks`. Looks for new disks and adds the correct `/dev` links

While your system is booting, notice references to the new disk, which will look roughly like the following:

```
sd1 at esp0: target 1 lun 0
sd1 is
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0
  <SEAGATE-ST15230N-0638 cyl 3974 alt 2 hd 19 sec 111>
```

This is a second indication that the disk is recognized and you will be able to access it when the system is fully booted. Most disks that you are likely to purchase will be preformatted and already partitioned. The question is, of course, whether they are partitioned to your liking. Usually, new disks are partitioned as if they were going to contain all the ordinary file systems for a Unix host. If you want to use a disk as one big file system, you don’t need to change anything. Simply use *slice 2* (once known as *partition c*), which represents the entire disk. Given the size and price of disks today, you might easily wind up with a 9 GB or 18 GB file system! You should consider the intended content and how you intend to back up file systems of this size before you choose to go this way. A decision to build a huge file system that might be filled to capacity should be partnered with a commitment to a digital linear tape (DLT) or other high-capacity backup device.

Formatting and Partitioning

Partitioning disks into usable chunks prevents one file system from spilling over into another. This can be a good thing if you’re interested in protecting one file system from others. It’s also a good thing if your backup strategies are different—some file systems you will want to back up much more frequently than others. On the other hand, breaking your disk into separate buckets costs you some amount of flexibility. You might easily wind up with excess space in one file system, while another fills up.

Following Standard Conventions

In Solaris, there are strong conventions about which slice is used for what, but no hard and fast rules. Unless there is an overriding reason not to do so, we always recommend following standard conventions. Following conventions is a time saver in the long run. Should someone else at some point try to recover data from one of the disks that you set up using another system, they will be more confident working with a disk that is laid out in a familiar way. At the same time, we see an increasingly large population of system administrators who prefer using a single large partition for everything. The advantage is that they are likely to never run out of space in one file system while space is wasted in another. The danger is that the root file system will fill up and the system become unusable. Should you want to employ the new single-partition convention, we suggest that you consider isolating any file system that might be susceptible to extremely large log files. We have seen, for example, application log files that have grown larger than 2 GB without anyone noticing until the `no room left on device` messages appeared. If you have applications that may create large log files such as this, we suggest using a separate partition (probably `/opt` or `/var`) so that the damage is contained.

The standard layout for Solaris partitions is shown in Table 1.5.

Physically, disks are composed of platters and heads that hover over the spinning platters in order to read and write. Accessing any particular file, therefore, depends on how much one of these heads has to move and how much disk I/O is backed up. Access time is composed of several things: how fast the disk is spinning, how far the head moves on average, and so on.

Partitioning with the format Command

Partitioning drives is done with the `format` command. You must be root to run it.

WARNING **Be careful if you are working on a live system. One of the authors once mistyped a disk address—the intended drive was one character different from the one she wound up using—and blew a file system out from underneath a dozen or more busy users. Along with the privilege of being root goes the possibility of making The Big Mistake. Double-check every choice before you hit Enter.**

Table 1.5 Standard Layout for Solaris Partitions

PARTITION	USE
0	Root (bootable)
1	Swap (`/tmp`)
2	The entire disk
3	`/var`
4	`/usr/local` (could be `/usr/openwin`)
5	`/opt`
6	`/usr`
7	`/export/home`

The `format` command is usually invoked simply with the command alone, but you can also include the disk device as an argument (e.g., `format /dev/rdsk/ c0t1d0s2`). Note that it is the raw device that is specified:

```
# format /dev/rdsk/c0t1d0s2
    selecting /dev/rdsk/c0t1d0s2
    [disk formatted]
    FORMAT MENU:
            disk       - select a disk
            type       - select (define) a disk type
            partition  - select (define) a partition table
            current    - describe the current disk
            format     - format and analyze the disk
            repair     - repair a defective sector
            label      - write label to the disk
            analyze    - surface analysis
            defect     - defect list management
            backup     - search for backup labels
            verify     - read and display labels
            save       - save new disk/partition definitions
            inquiry    - show vendor, product, and revision
            volname    - set 8-character volume name
            quit
```

If you enter **format** at the next prompt, you'll wind up doing a low-level format on the disk. This takes a lot of time and is almost never needed. Only if you are using a drive that you suspect may have problems should you bother with the formatting, analysis, and repair options provided with the `format` command.

The next step is to repartition the drive. To do so, type **partition** at the prompt. The following menu will appear:

```
format> partition
PARTITION MENU:
        0      - change `0' partition
        1      - change `1' partition
        2      - change `2' partition
        3      - change `3' partition
        4      - change `4' partition
        5      - change `5' partition
        6      - change `6' partition
        7      - change `7' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
```

To view the existing partition configuration, enter **print**.

```
      partition> print
Current partition table (original):
Total disk cylinders available: 8152 + 2 (reserved cylinders)
Part      Tag    Flag     Cylinders         Size                  Blocks
0        root    wm        0 -   59      128.32MB    (60/0/0)     262800
1        swap    wu       60 -  179      256.64MB    (120/0/0)    525600
2      backup    wu        0 - 8151       17.03GB    (8152/0/0) 35705760
3       stand    wm      180 - 6975       14.19GB    (6796/0/0) 29766480
4        home    wm     6976 - 7487        1.07GB    (512/0/0)   2242560
5 unassigned    wm     7488 - 7743      547.50MB    (256/0/0)   1121280
6         usr    wm     7744 - 7999      547.50MB    (256/0/0)   1121280
7         var    wm     8000 - 8031       68.44MB    (32/0/0)     140160
```

If you want to change the partitioning on this drive, you should be careful to preserve the continuity of the cylinders. Note how the ranges follow in sequence: 0-59, 60-179, 180-6975, and so on. If you alter the size of one partition, adjust the adjoining partitions accordingly and print the partition table again to be sure that the partitions still line up. To adjust a partition, enter its partition number and respond to the prompts. Here, we're combining two partitions into one and assigning the space to partition 6:

```
partition> 5
Enter partition id tag[unassigned]:
Enter partition permission flags[wm]:
Enter new starting cyl[7488]:
Enter partition size[1121280b, 256c, 547.50mb]: 0c
partition> 6
Enter partition id tag[usr]:
Enter partition permission flags[wm]:
Enter new starting cyl[0]: 7488
Enter partition size[1121280b, 256c, 547.50mb]: 512c
```

Note that the space can be allocated in blocks, cylinders, or megabytes. We prefer to work in cylinders. When you've finished partitioning the disk and are sure your partition map makes sense, write the label back to the disk:

```
partition> label
Ready to label disk, continue? y
```

You don't have to be using the `format` command to view the partition table. You can also list it with the `prtvtoc` command (at the normal Unix prompt), as shown here:

```
spoon# prtvtoc /dev/dsk/c0t0d0s2
* /dev/dsk/c0t0d0s2 partition map
*
* Dimensions:
*     512 bytes/sector
*     219 sectors/track
*      20 tracks/cylinder
*    4380 sectors/cylinder
*    8154 cylinders
*    8152 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
* Unallocated space:
*       First     Sector    Last
*       Sector    Count    Sector
*    35180160    521220  35701379
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount
Directory
       0      2    00          0    262800    262799
       1      3    01     262800    525600    788399
       2      5    01          0  35705760  35705759
       3      6    00     788400  29766480  30554879
       4      8    00   30554880   2242560  32797439
       5      0    00   32797440   1121280  33918719
       6      4    00   33918720   1121280  35039999
       7      7    00   35040000    140160  35180159
```

TIP You can use the `fmthard` command to put a partition table on a disk and avoid the `format` command. If you have several disks with the same layout, this can save you time. Try using `prtvtoc` to get the volume table of contents (vtoc) that you want to copy.

Creating New File Systems

After you've partitioned your new disk to your liking, it's time to build new file systems. You need to build a file system before you can use a partition (unless it's to be used as a raw partition). A certain percentage of the overall space will be used for overhead—the inodes, free-block lists, block maps, superblocks, and space reserved for file system elbow room (free space), because an absolutely full disk would be impossible to work with. As file systems become too full, they slow down. Space becomes harder to allocate, fragmentation increases, and file activity is likely to be higher. See the material on monitoring your environment in Chapter 9.

When preparing to create a file system, there are a number of decisions that you can make that might significantly influence its performance. These include the block size, ratio of disk space to inodes, and free space—all of which are parameters that can be specified with the `newfs` command. The `newfs` command is, in effect, a front end to `mkfs` that supplies values for many of the command's options.

Table 1.6 newfs Parameters

PARAMETER	PURPOSE
a	Alternate blocks per cylinder (default is 0).
b	Block size, in bytes (default is 8192).
c	Cylinders per cylinder group (default is 16).
I	Amount of disk space to reserve, in bytes, for each inode created (default is 2048).
f	Size of a fragment, in bytes (default is 1024).
m	Percentage of free space (10 percent on small disks, less on larger disks).
o	Optimization method, space or time (default is time).
r	Rotational speed of the disk, in revolutions per minute (default is 5).
s	Size of the file system, in blocks (default is 2880).
t	Tracks per cylinder (default is 2).
v	Verbose—display parameters passed to mkfs (not used by default).
N	No change—don't create file system, but provide parameters that would be used (not used by default).

At a minimum, you have to tell `newfs` where to build the file system. A command like `newfs /dev/rdsk/c0t2d0s2` would create a file system using all the space on the specified disk. For values for parameters different from the default, use the options shown in Table 1.6.

Here is an example of a `newfs` command that specifies one inode for each 8K of disk space and reserves 1 percent of the overall space for free space:

```
newfs -i 8192 -m 1 /dev/rdsk/c0t2d0s2
```

The ratio of disk space reserved for inodes to disk space reserved for files in this case is 1 to 16. This is because inodes are roughly, if not precisely, 512 bytes in size. Because the default is 1 to 4, this shows the user is expecting larger files than is usually the case.

If you specify the -N and -v options with `newfs`, you'll see output that details the resulting `mkfs` command and the parameters that would be used in creating a file system *without* creating it. Similarly, the `mkfs` command with a -m parameter reveals the parameters of an existing file system, as shown here:

```
# mkfs -m /dev/rdsk/c0t2d0s2
mkfs -F ufs -o nsect=80,ntrack=19,bsize=8192,fragsize=1024,
cgsize=32,free=4,rps=90,nbpi=4136,opt=t,apc=0,gap=0,nrpos=8,
maxcontig=16 /dev/rdsk/c0t2d0s2 4023460
```

Note that the mkfs command includes a parameter for the type of file system. The newfs command defaults to UFS.

UFS supports block sizes up to 64K. The block size is specified when a file system is created. In addition, a smaller unit of space—the fragment—is created to allow for more efficient data storage within the last block of a file. Each block can hold a number of segments (1, 2, 4, or 8) fragments. Transfer is still done in blocks. Superblock replication makes the file system more resistant to failure. Also, placement of superblocks and inodes is such that it is unlikely that damage to the file system would destroy so much data that the file system would not be largely recoverable.

A *raw partition* does not contain a file system at all. Instead, the application using it (often a data base) keeps track of where the data is on the disk. This kind of access can be faster, because updates to metadata and block-mapping structures are not required. On the other hand, a raw partition is not general purpose; only the controlling application knows how to access the data it contains.

The newfs command provides a simplified interface to the mkfs command. In the following command, we're building a file system on partition 5, specifying less than the normal percentage of free space. There are numerous other parameters that you can specify (especially if you deal directly with mkfs) to tune your file systems, but you should not use these options without considerable knowledge of how the file system will be used and the implications of each parameter.

```
spoon# newfs -m 4 /dev/rdsk/c0t0d0s5
setting optimization for space with minfree less than 10%
newfs: construct a new file system /dev/rdsk/c0t0d0s5: (y/n)? y
/dev/rdsk/c0t0d0s5:     1121280 sectors in 256 cylinders of 20
tracks, 219 sectors
        547.5MB in 16 cyl groups (16 c/g, 34.22MB/g, 16448 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 70336, 140640, 210944, 281248, 351552, 421856, 492160, 562464,
632768,
 703072, 773376, 843680, 913984, 984288, 1054592,
...
```

Some file system tuning parameters can be adjusted at any time, but only a few. The best source of advice on file system and general performance tuning is, of course, Adrian Cockcroft, who writes for Sun and for *SunWorld* magazine. The second edition of his *Sun Performance and Tuning* book (with Richard Pettit) is included in the Bibliography and Recommended Reading list.

Mounting and Unmounting

Sysadmins used to do a lot of mounting and unmounting of file systems; the mount and umount commands are, after all, privileged: only the superuser can issue them to mount or unmount a file system. Then automounter and vold came along, relieving sysadmins of having to be present and involved in every mount and, in the case of automounter, bringing considerable improvement in NFS performance at the same time.

With *automounter,* reference to a mountable file system results in its being mounted. A cd to a directory or an ls of its contents causes a file system to mount—provided the user issuing the command has proper permissions for the mount point, and the client has permission to mount the file system in question. A *mount point* is simply a directory. However, those mount points that automount uses are reserved; you cannot, for example, mount a file system on /home when automounter is running and the mounting of home directories is within its purview. *vold* automatically mounts file systems on diskettes or CD-ROMs when they are inserted into the drive—provided, of course, that vold is running (it is by default). The file systems unmount and eject the media if the eject command is issued—provided that no one is using the mounted files; simply having cd'ed into a mounted directory occupies it.

The Life Cycle of Mount Operations

Whenever a system is booted, file systems are mounted. Some of these file systems are mounted early in the boot process; the system would not be able to boot without them. You may notice in the /etc/vfstab file shown a bit later in this section that neither the root nor the /usr file system is specified to be mounted at boot time. These file systems house the files that direct the process of booting (described in Chapter 4) and must be available soon after the basic hardware checks. Other file systems are mounted later in the boot process, as configured in the /etc/vfstab file. Still others are mounted only as requested via a mount command or, as previously mentioned, in response to requests via automounter.

The /etc/vfstab file specifies file systems to be mounted along with locations and options. It is read during the boot process by the start script /etc/rc2.d/mountall.

The file itself is a white-space-delimited list of files with columns specifying the following:

- The file system to be mounted
- The corresponding raw device
- The mount point
- The file system type
- The fsck order
- Whether to mount at boot and with the mountall command
- Options—other mounting options (e.g., readonly)

Here is a sample /etc/vfstab file:

```
#device           device             mount   FS     fsck mount   mount
#to mount         to fsck            point   type   pass at boot options
/proc             -                  /proc   procfs -    no      -
fd                -                  /dev/fd fd     -    no      -
swap              -                  /tmp    tmpfs  -    yes     -
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 /       ufs    1    no      -
```

```
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /usr    ufs    2    no      -
/dev/dsk/c0t3d0s5 /dev/rdsk/c0t3d0s5 /opt    ufs    5    yes     -
/dev/dsk/c0t3d0s1 -                  -       swapfs -    no      -
```

NFS server support is not provided (by default) until run state 3 (discussed in Chapter 4). This means that systems do not export their file systems until they have moved into this run state, commonly referred to as *multiuser with NFS.* Refer to the file `/etc/rc3.d/S15nfs.server` to see how the NFS processes are started for an NFS server. The client script `/etc/rc2.d/S73nfs.client` allows the mounting of remote file systems in run state 2 (this is also effective in run state 3, as we shall explain in Chapter 4).

The file that controls which file systems are shared by a host is `/etc/dfs/dfstab`. In the following sample `dfstab` file, we're using a `netgroup` rather than a string of host names. Refer to the `netgroups` man page for more information on `netgroups`.

```
#    Place share (1M) commands here for automatic execution
#    on entering init state 3.
#
#    share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
#    .e.g.,
#    share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2
share -F nfs -o rw=utensils -d "apps" /apps
share -F nfs -o rw=utensils -d "archives" /archives
share -F nfs -o rw=utensils -d "home" /raid/home
share -F nfs -o ro=utensils -d "cdrom" /cdrom/cdrom0
```

File systems of type UFS repeat superblock information at various intervals within a file system. These structures contain file system statistics (e.g., the number of files, the label, the overall size, etc.). If the initial superblock is damaged, `fsck` gives you an opportunity to provide another `reboot -n` after an `fsck`; avoid the sync so you don't overwrite your fixes by flushing the output buffers.

The Automounter

Automounter is a standard Solaris feature that streamlines the management of shared file systems and reduces network traffic. Automounter also helps avoid file system problems in environments with many servers and many more clients by timing out on file system mounts instead of freezing one system's resources when a server from which it has mounted a file system becomes unavailable. In addition to the material presented here, we suggest that you refer to anything written by Hal Stern (see Bibliography and Recommended Reading) regarding NFS and automounter to get a more thorough picture than we can present here.

Automounter uses centrally managed configuration information to specify the file systems and mount points for a network. By doing so, it helps to enforce a consistency in the overall naming and arrangement of file systems. It reduces the overhead of NFS mounts by dismounting file systems after a period of nonuse (the default is 5 minutes). Automounter is one of the busy sysadmin's best friends—it provides what a user needs, when a user needs it, and then it goes away by itself. It dramatically reduces

NFS hangs and is fairly easy to administer. File systems that are automounted don't need to be specified in each potential client's `/etc/vfstab` file. In fact, all the client system needs is the mount point (created by `automountd`), and a reference to the file system will cause it to be mounted. The rest is accomplished through NIS maps or NIS+ tables—or their file equivalents, but this is rarely the case.

The services provided by automount are actually brought about through the efforts of several components—the automount maps (detailing what gets mounted, where, and how), the user-level process that acts like an NFS daemon, and the kernel module, `autofs`, that invokes mounts and dismounts.

The daemon process, `automountd`, runs on clients. RPC requests are generated in response to requests for file systems as routine as a user changing directory (i.e., using `cd`) into a directory managed by automounter. Once `automountd` intercepts a request for data, it passes the request to the NFS server and steps aside (there would be a performance hit if automounter got involved in every transfer of data). After 5 minutes of inactivity, however, automounter sends an NFS request for a dismount.

On large and frequently changing networks, there are few tools that will save you as much headache as the automounter. The days of manually editing files on numerous hosts (e.g., the `/etc/vfstab` files) are over with the centralized management made possible with automounter. Not only does automounter make it possible for users to log in virtually anywhere on your network and have their familiar home environment available to them, it ensures that file systems unmount when not in use and reduces the network traffic related to NFS. A third advantage is that automounter keeps a system from hanging if the server from which it is automounting file systems is not available.

Although automounter provides users with automatic mounting, there is some setup required on the part of the sysadmin—a proper configuration of the automount maps and sharing entries on the systems from which these file systems derive.

Security and Automounter

Automounter is subject to the same security considerations as NFS. Even though the tool provides a convenient way to automate mounts, it still requires basic access privilege. Regardless of what automount maps may dictate, a remote file system cannot be mounted if it is not shared and cannot be mounted if access to the mount point is not available.

Automounter Configuration Files

Automounter uses one or more files—usually two or three—to control which file systems are mounted and when. The `/etc/auto_master` file details the other files and maps that are used. The other two maps that you'll generally find in use are `/etc/auto_home` and `/etc/auto_direct`. The existence and configuration of these files depends on how automounter is used at a particular site.

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

In this particular /etc/auto_master file, the /net entry tells automounter (i.e., automountd) to manage the mounting of file systems in the /net directory. Any file systems exported and available to the local system can be accessed via the /net directory. The /home entry instructs automounter to manage home directories using the information found in the /etc/auto_home file. The third entry, starting with /-, calls the /etc/auto_direct map into use. We'll discuss direct maps in just a moment.

```
# Home directory map for automounter
#
#+auto_home
*     -rw  Rwanda:/raid/home/&
```

Entries in /etc/auto_home of the type user -nosuid host:/home/user cause an individual's home directory to mount when requested. An entry of the form * -nosuid server:/home/&, on the other hand, causes any home not previously specified to be mounted from the system server. The nosuid option prevents suid program execution from the mounted file system.

The third commonly used automounter file that you will encounter is auto_direct. This file is used for direct maps. The next section briefly examines the difference between direct and indirect maps and the implications of these differences on your network.

Direct and Indirect Maps

Automounter uses two types of maps to describe the mounting of file systems and directories onto mount points. They are called direct maps and indirect maps. For file systems using *direct maps,* the mounts occur on a directory (e.g., /export/home is mounted on /home). For file systems using *indirect maps,* the mounts occur within a directory (e.g., /export/home/sandra is mounted on /home/sandra). Figure 1.6 displays the result of both direct and indirect maps. Automounter knows which file systems it is required to watch. It can be said to own certain mount points and to intercept requests when they are made.

Here is an example of a direct map. The lines in this particular map direct automounter to use webserver:/usr/WWW whenever anyone refers to the directory /WWW, the intranet:/usr/intranet directory when anyone refers to /intranet, and mailserver:/var/mail when anyone refers to /var/mail.

```
# direct directory map for NFS mounts
#
/WWW      webserver:/usr/WWW
/intranet intranet:/usr/intranet
/var/mail -actimeo=0 mailserver:/var/mail
^         ^
|         |
key       value
```

An indirect map might look like this:

```
evan      server:/export/home/evan
sandra    server:/export/home/sandra
```

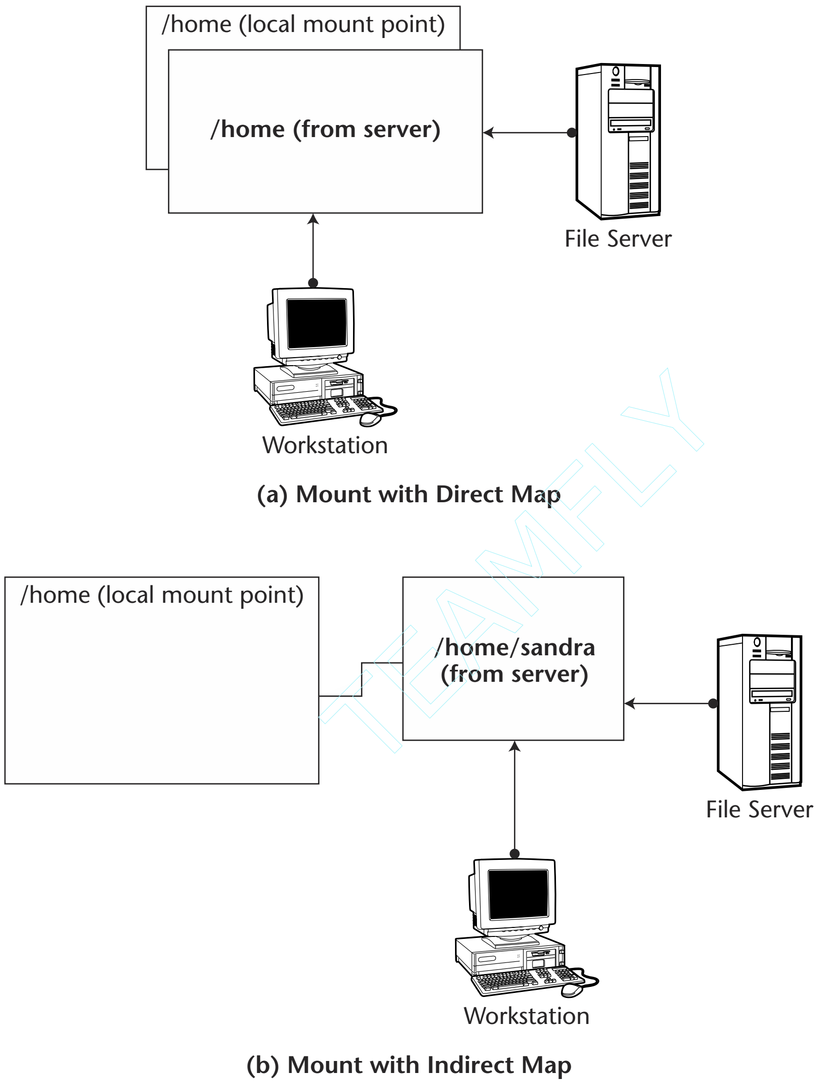

/home (local mount point)
/home (from server)
File Server
Workstation
(a) Mount with Direct Map
/home (local mount point)
/home/sandra (from server)
File Server
Workstation
(b) Mount with Indirect Map

Figure 1.6 Direct and indirect maps.

The preceding would mount the file systems `/export/home/evan` and `/export/home/sandra` on `/home/evan` and `/home/sandra` (these entries appear in the auto_home map). A more general solution is to use an entry such as the following one, which would cause the home directory for any user from the file server `server` to be mounted as `/home/username`.

```
server:/export/home/&
```

This causes each user's home directory to mount on `/home/user`, and so on.

Direct maps are a lot like indirect maps, but are more explicit. Each direct map entry provides a one-to-one mapping of source and mount point, much like a traditional mount. Whereas with indirect maps, more general-purpose associations between mount points and directories are managed by the automounter using the entries both literal and wildcarded from the `/etc/auto-*` files, direct maps specify direct relationships.

Direct maps can also be of the form `/filesystem  host1,host2,host3:/filesystem`. When this is the case, any of the hosts listed can supply the file system when a request for an automount is made. Whichever system responds first provides the mount—an easy way to load-balance and reduce reliance on any single server, without a lot of extra effort. You can also add weighting factors in parentheses (the higher the better) to influence this process. An `auto_direct` entry of `/apps server1(50),server2,server3(10):/apps` would encourage your systems to mount the `/apps` directory from `server1`.

Centralizing Services

For most networks, centralizing mail services makes a lot of sense—you only have to configure one server `sendmail` file and you can keep mail files in a single place where they are easy to manage. The central location also facilitates removal of lock files or pop files. You probably will want to provide a single mail server that corresponds to MX record in DNS; individual hosts can get mail by mounting `/var/mail`, or by using services such as POP3 or IMAP4.

With the low-price and large disks often available on PC and Macintosh systems, it might make sense to limit the use they make of network resources. Some server resources, on the other hand, can facilitate file sharing without requiring users to use `ftp` or `fetch` to transfer files back and forth, an extra step.

How do you decide what to mount and what to automount? One of the clear advantages of automounting is that it's all driven from maps that you create and administer centrally. This is in line with what we've been saying all along about the ideal configuration for sysadmins who want to go home at night: avoid complexity—and when you can't, keep it where you can closely watch it. More importantly, the network load is considerably decreased with automount, and you won't suffer from system hangs when systems or file systems are unavailable (see Figure 1.7).

As a general rule, it is best to avoid hard-mounting remote file systems on NFS servers. To do so is complicated and somewhat risky (it could lead to a deadlock situation in which each of two servers is waiting for the other). However, it isn't necessarily obvious how to avoid mounts if users Telnet to your file servers, because their home directories will be automounted. One of the authors, in fact, once saw a file server run out of swap space while trying to mount home directories from another server that had crashed. Though no one was logged in at the time, the server with the mount problem was a mail server, and it was attempting to automount a home directory each time a piece of mail arrived. The necessity to check for `.forward` files in the users' home directories required this. As the number of hung `sendmail` processes skyrocketed, swap space was soon exhausted. Codependency problems of this sort can be avoided, but only with careful assignment of server roles and an understanding of the processes involved.

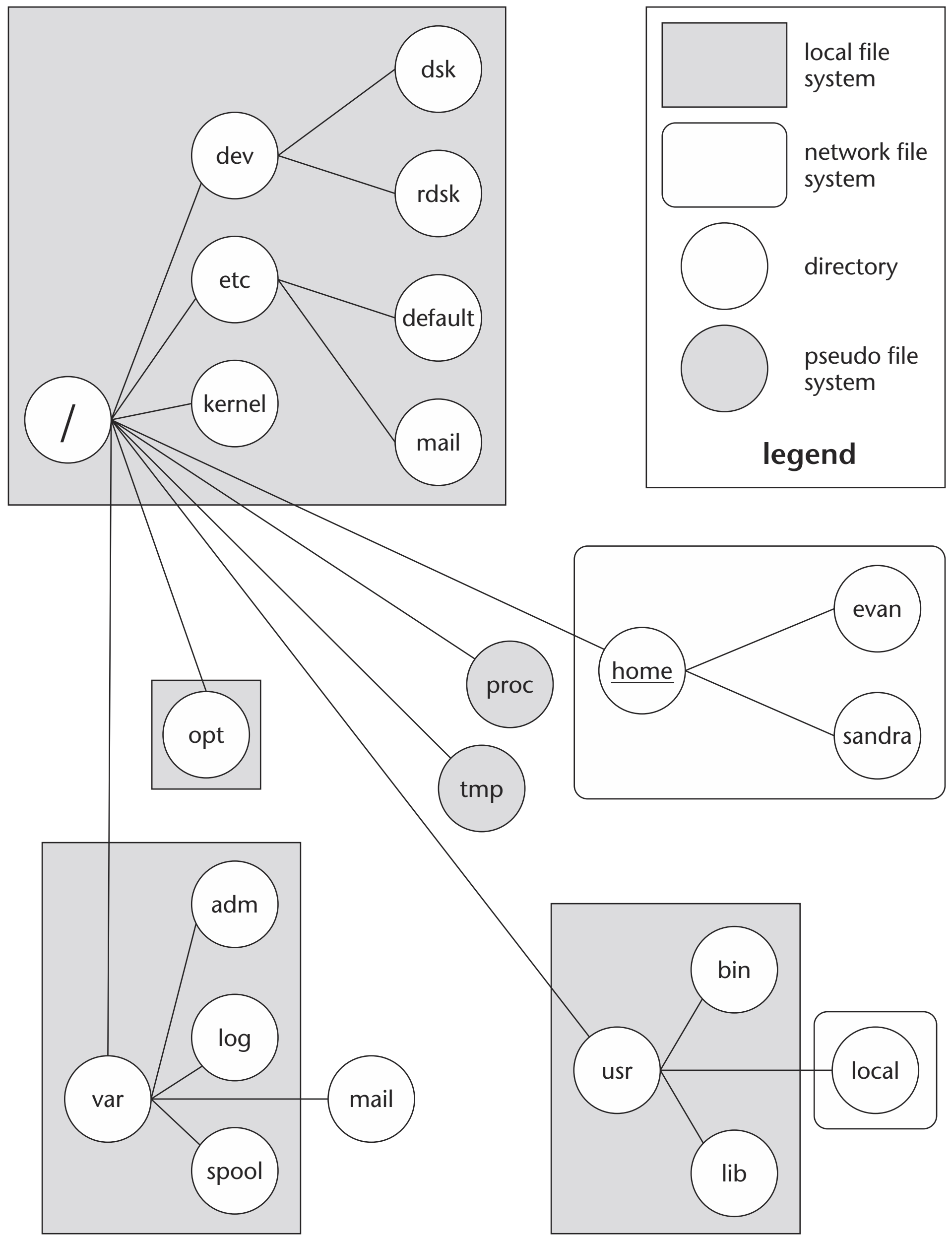


Figure 1.7 File systems of a typical Solaris client.

TIP A tidy solution to the problems mentioned in the previous paragraph is to configure `sendmail` to look somewhere else for `.forward` files. This feature is available in current releases of `sendmail` (look for the `ForwardPath` variable). Another option is to run mail and home directory services on the same machine, avoiding the requirement for cross-mounting and its potential consequences.

NFS starts up only when a system boots or enters run state 3—if and only if there are file systems to be shared. How does the system know? Easy—it looks at the file `/etc/dfs/dfstab`. If there are file systems listed in the file, NFS will share them. The `/etc/rc3.d/nfs.server` script starts NFS.

NFS file systems should always be shared using host lists—and should never be available to the world in general. To share file systems with no access restrictions introduces a security risk. File systems can be shared as read-only (as should always be the case with CD-ROMs unless you want to encounter a lot of errors). The most common mode of sharing file systems is to include a list of hosts and give them read/write access, like this:

```
share -o rw=congo,rwanda,malawi
```

Only under extreme circumstances should you allow another host to have the same authority on a file system as the local root. To allow this, use the following syntax:

```
share -o rw=congo,rwanda,malawi root=zimbabwe
```

If you are supporting netgroups (available only with NIS or NIS+), you can often save yourself a lot of trouble by using them in place of lists of hosts. Consider, for example, the following `/etc/dfs/dfstab` file:

```
#This is the dfstab file.
share /home -o
rw=congo,rwanda,malawi,chad,zaire,sudan,burundi,kenya,zimbabwe
share /usr/local -o
rw=congo,rwanda,malawi,chad,zaire,sudan,burundi,kenya,zimbabwe
```

This file could be reduced to the following more approachable file:

```
#This is the dfstab file.
share /home -o rw=africa
share /usr/local -o rw=africa
```

You would define the netgroup africa in the /etc/netgroup file. To view the file systems that are shared by any system, simply use the `share` command, as shown here:

```
congo% share
- /usr/man  ro=africa   "man pages"
- /usr/WWW  rw=tonga:botswana:malawi:gabon:zaire:burundi "WWW"
```

The NFS mount options are described in Table 1.7.

Table 1.7 NFS Mount Options

OPTION	DESCRIPTION
rw	Read-write
ro	Read-only
root	Root access enabled
suid	setuid permitted
nosuid	setuid not permitted

Automounter Variables

As stated earlier, automounter is increasingly useful the more systems you manage and the more your users move from one system to another, whether physically or using remote login commands. To simplify maintenance of your automount maps, the automounter provides a number of variables that take the place of literal entries. The variable ARCH, for example, represents the architecture of a particular machine. OSREL, on the other hand, specifies the operating system (OS) release. If you were mounting `/usr/lib` from a remote system, you could more easily and more *generically* specify file systems to be mounted by using these variables. They are listed and briefly explained in Table 1.8.

Table 1.8 Automounter Variables

VARIABLE	DESCRIPTION	AS DERIVED FROM	EXAMPLE
ARCH	Application architecture	uname -m	sun4m
CPU	Processor type	uname -p	sparc
HOST	Host name	uname -n	spoon
OSNAME	Operating system name	uname -s	SunOS
OSREL	Operating system release	uname -r	5.9
OSVERS	Operating system version	uname -v	Generic
NATISA	Native instruction set	isainfo -n	sparc

Options for NFS

While the Network File System (NFS) is heavily used on some sites and completely avoided at others, there are some options that might make it both more reliable and more secure. In this section, we describe optional logging as some options for making NFS more secure.

NFS Server Logging

NFS logging, available through the activity of the NFS log daemon (`nfslogd`), logs NFS activity on the NFS server. The generated log files contain information about the files being accessed, the client system, the direction of the file activity (input or output from the perspective of the server), and the number of bytes moved. This format can be extended to include directory modification operations, such as creating or removing a directory.

NFS logging is new with Solaris 8 and generally used by sites offering file downloads via NFS or WebNFS to track activity. NFS log files can get large, so it is not a good idea to collect this information unless it will be used. At the same time, log files are rotated and a specified number of files are retained, limiting the overhead associated with the logging operation.

The option for logging must be specified when a file system or directory is shared.

How Does NFS Logging Work?

The NFS log daemon generates logging information by analyzing RPC activity associated with NFS. There are two phases to logging, one in which the basic activity (i.e., raw RPC requests) is recorded in work buffers and another in which the contents of these work buffers is examined, grouped, and analyzed, and an output log generated.

The NFS log daemon sleeps after the analysis is complete, while additional information is logged. The duration of this sleep is configured in the configuration file with the `IDLE_TIME` parameter.

The NFS log daemon also maintains a mapping of file handles to pathnames so that it will be able to create reports that will be meaningful to those who read them later. After all, file handles are temporary and not particularly human-friendly. This mapping is updated any time a client performs a lookup. To ensure the resiliency of NFS, this mapping file (also referred to as a mapping “database”) is also stored on disk—in a location determined by the configuration file—so that the associations between file handles and paths can survive a reboot of the server. The mapping file is called `fhtable` by default.

Enabling NFS Logging

NFS logging requires very little setup. If you are satisfied with the default configuration, all you need to do is select which shared file systems you want to log, specify the logging, and restart sharing with logging enabled. For example, a normal share might look like this in your `/etc/dfs/dfstab` file:

```
share -F nfs -d "ftp site" /export/home/ftp
```

When you share the file system with the `share` command, your choices are reflected in the share command output:

```
# share
-               /export/home/ftp   rw   "ftp site"
```

To add NFS logging, you would add the `log=tag` line as shown in the following lines. In the first example, the `log` option is the only option. In the second, the `log` option is used along with the `read-only` specification. Use `log=global` unless you have added a tag of your own to the `/etc/nfs/nfslog.conf` file.

```
share -F nfs -o log=global -d "ftp site" /export/home/ftp
share -F nfs -o ro,log=global -d "ftp site" /export/home/ftp
```

You would then share the file system or directory again, as follows:

```
# share
-               /export/home/ftp   log=global   "ftp site"
```

If this is your first shared file system, a new file (i.e., `sharetab`) will be added to the `/etc/dfs` directory:

```
# ls -l
total 6
-rw-r--r--   1 root     sys          451 May 27 13:03 dfstab
-rw-r--r--   1 root     root          68 Jan 20 20:33 fstypes
-rw-r--r--   1 root     other         43 May 27 13:03 sharetab
```

You might notice that a new file (`nfslogtab`) has also appeared in your `/etc/nfs` directory:

```
# ls -l
total 4
-rw-r--r--   1 root     sys          388 Jan 20 20:33 nfslog.conf
-rw-r--r--   1 root     other         53 May 27 13:03 nfslogtab
```

This file displays information about the ongoing logging just as `mnttab` tells you about currently mounted file systems and `sharetab` tells you about file systems that are currently shared. The contents of this file will look something like this:

```
# cat nfslogtab
/var/nfs/nfslog_workbuffer     /export/home/ftp      global  1
```

The /var/nfs/nfslog_workbuffer string identifies where the logging information is kept, though the contents of the /var/nfs directory actually look more like this:

```
# cd /var/nfs
# ls -l
total 2
-rw-------   1 root     other        244 May 27 13:03
nfslog_workbuffer_log_in_process
# file nf*
nfslog_workbuffer_log_in_process:       data
```

If you haven't already started NFS server support, no one will be able to mount the file systems that you are sharing. Instead, they will see output similar to the following:

```
# mount dragonfly:/export/home/ftp /ftp
nfs mount: dragonfly: : RPC: Program not registered
nfs mount: retrying: /ftp
```

All you need to do to rectify this situation is to start the services by running the start script in /etc/init.d:

```
# /etc/init.d/nfs.server start
```

Then the mount works as intended:

```
# mount dragonfly:/export/home/ftp /mnt
```

After NFS logging is started, you will soon see activity in the logging directory. Notice that the files in your /var/nfs directory are growing larger.

```
# ls -l
total 58
-rw-r-----   1 root     other       4096 May 27 13:19
fhtable.0074008400000002.dir
-rw-r-----   1 root     other      28672 May 27 13:19
fhtable.0074008400000002.pag
-rw-r-----   1 root     other       3653 May 27 13:19 nfslog
-rw-------   1 root     other         24 May 27 13:19
nfslog_workbuffer_log_in_process
```

If you want to change your logging settings, edit the /etc/default/nfslogd file. This file allows you to modify any of five settings. To change a setting, remove the comment character (the value displayed is the default), change the value to the value you want, and restart the daemon.

The parameters that you can change and their default values are listed in Table 1.9. The descriptions shown are included in the file.

Table 1.9 nfslogd Parameters

PARAMETER	DEFAULT	DESCRIPTION
MAX_LOGS_PRESERVE	10	Maximum number of logs to preserve
MIN_PROCESSING_SIZE	524288	Minimum size buffer should reach before processing
IDLE_TIME	300	Number of seconds the daemon should sleep waiting for more work
CYCLE_FREQUENCY	24	The frequency (in hours) with which the log buffers should be cycled
UMASK	0137	UMASK to use for the creation of logs and file handle mapping tables

The contents of the log file, once it has been created from the raw data, will look like this:

```
Mon May 27 13:12:29 2002 0 bullfrog.dragonflyditch.com 10550
/export/home/ftp/roles0.gif b _ o r 60001 nfs 0 *
Mon May 27 13:13:54 2002 0 bullfrog.dragonflyditch.com 48747
/export/home/ftp/smc0.jpg b _ o r 60001 nfs 0 *
```

It is easy to pick out the date/time stamp in this file. The name of the client (i.e., `bullfrog`) and name of the individual files (e.g., `/export/home/ftp/smc17.jpg`) are also included. The operation in each of these sample lines is “o” for output. This indicates that the client retrieved the contents of these files. The file sizes vary from 10,550 to 77,852 bytes. The column containing `60001` indicates the UID of the user on the client.

NFS Logging Files

NFS logging involves the use of a small number of files. Each of these files and directories is outlined in the following sections.

/usr/lib/nfs/nfslogd

This is the log daemon. It is generally started when your server is booted (through the `/etc/rc3.d/S15nfs.server` script and the presence of the log option in the `/etc/dfs/dfstab file` (see the “/etc/dfs/dfstab” section a bit later in this chapter).

/etc/nfs/nfslog.conf

The `nfslog.conf` file is the basic configuration file for NFS logging. It specifies the directory in which log files will be stored, the name of the NFS mapping file, and the preface for work buffers. You can change the options in this file if you want to change your NFS logging settings on a systemwide basis or you can add a new tag with different parameters.

```
#ident "@(#)nfslog.conf 1.5 99/02/21 SMI"
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
# All rights reserved.
#
# NFS server log configuration file.
#
# [ defaultdir= ] \
#     [ log= ] [ fhtable= ] \
#     [ buffer= ] [ logformat=basic|extended ]
#

global defaultdir=/var/nfs \
log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
```

/etc/default/nfslogd

This file establishes the defaults for the NFS log daemon.

```
#
#ident      "@(#)nfslogd.dfl    1.8  99/02/27 SMI"
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
# All rights reserved.
#

# Specify the maximum number of logs to preserve.
#
# MAX_LOGS_PRESERVE=10

# Minimum size buffer should reach before processing.
#
# MIN_PROCESSING_SIZE=524288

# Number of seconds the daemon should sleep waiting for more work.
#
# IDLE_TIME=300

# CYCLE_FREQUENCY specifies the frequency (in hours) with which the
# log buffers should be cycled.
#
# CYCLE_FREQUENCY=24

# Use UMASK for the creation of logs and file handle mapping tables.
#
# UMASK=0137
```

/etc/dfs/dfstab

This file specifies sharing operations. This is where you insert the option to enable NFS logging on a particular file system or directory.

```
#    Place share(1M) commands here for automatic execution
#    on entering init state 3.
#
#    Issue the command '/etc/init.d/nfs.server start' to run the NFS
#    daemon processes and the share commands, after adding the very
#    first entry to this file.
#
#    share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
#    .e.g,
#    share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2

share -F nfs -o log=global -d "ftp site" /export/home/ftp
```

/etc/dfs/sharetab

This file contains information on currently shared file systems. The `share` command reads this file.

```
/export/home/ftp   -     nfs     log=global     ftp site
```

/etc/nfs/nfslogtab

This file contains information on current NFS logging activity.

```
/var/nfs/nfslog_workbuffer /export/home/ftp     global    1
```

/etc/rc3.d/S15nfs.server (aka /etc/init.d/nfs.server)

This is the start script that initiates NFS server processes when a server is rebooted. Though we won't include the entire file here, this excerpt contains the commands that cause file systems to be shared and start the NFS log daemon if the `log=` option is used in the `dfstab` file.

```
# If /etc/dfs/dfstab exists and has non-blank or non-commented-out
# lines, then run shareall to export them, and then start up mountd
# and nfsd if anything was successfully exported.

startnfsd=0
if [ -f /etc/dfs/dfstab ] && /usr/bin/egrep -v '^[   ]*(#|$)' \
    /etc/dfs/dfstab >/dev/null 2>&1; then

    /usr/sbin/shareall -F nfs

    if /usr/bin/grep -s nfs /etc/dfs/sharetab >/dev/null; then
        startnfsd=1
    fi
fi

# Start nfslogd if /etc/nfs/nfslogtab exists and is not empty.
# This means that either we just shared new filesystems with
# logging enabled, or they were shared in the previous session
# and their working buffers haven't been processed yet.

if [ -s /etc/nfs/nfslogtab ]; then
    /usr/lib/nfs/nfslogd &
fi
```

/var/nfs

This is the default directory for NFS log files and work buffers. Following is a sample listing. The `nfslog` file is an ASCII file.

```
-rw-r-----   1 root     other       4096 May 27 13:19
fhtable.0074008400000002.dir
-rw-r-----   1 root     other      28672 May 27 13:19
fhtable.0074008400000002.pag
-rw-r-----   1 root     other       3653 May 27 13:19 nfslog
-rw-------   1 root     other         24 May 27 13:19
nfslog_workbuffer_log_in_process
```

Securing NFS versus Secure NFS

There are a number of measures that you can take to improve the security of your NFS services. If your security concerns warrant, you should probably go to the trouble of running Secure NFS. However, you should be prepared to deal with the added complexity and be prepared for the lack of Secure NFS products for non-Solaris clients.

Following are some guidelines for running NFS. Refer to Chapter 12 for more information on NFS and security:

- Don't run NFS on any secure gateway.
- Share files only with known hosts or netgroups of hosts.
- Share file systems with the least privileges possible. If file systems can be exported read-only, do so. Use `nosuid`. Don't use `root`, unless you absolutely must.
- Don't forget automount maps.

Administering Secure NFS

Authentication is the process of validating a username and password entered at login. For Secure NFS, the authentication system is more rigid. You have a choice of Diffie-Hellman and/or Kerberos authentication, both of which use complex encryption to provide the validation information that supports the authentication process.

The commands to establish key sets for *Diffie-Hellman authentication* are the newkey and nisaddcred commands. Your users will have to establish their own secure RPC passwords. Once created, these keys are stored in the publickey database.

Verify that the name service is running. If you are using NIS+, the command to do this is nisping -u. This command will tell you if the service is up and when the last updates were made. It also will tell you about replica servers. For NIS, use ypwhich. It will tell you if ypbind is running and which NIS server the system is bound to.

The process that manages authentication keys is the keyserv process (/usr/sbin/keyserv). This needs to be running for secure NFS to work. Use keylogin to decrypt and store the secret key. If the login password is the same as the network password, you don't have to do this.

Add the option -o sec=dh to the share options. For example:

```
share -F nfs -o sec=dh /export/home
```

Add the entry to the auto_master map as well. For example:

```
/home auto_home -nosuid,sec=dh
```

NFS refuses access if the security modes do not match, unless -sec=none is on the command line.

The file /etc/.rootkey should be preserved across system upgrades and changes to ensure that the authentication continues to work. If you inadvertently destroy this file, you can create another with the keylogin -r command.

The process for *Kerberos authentication* involves updating the dfstab file and the auto_master map, as shown here:

```
# share -F nfs -o sec=krb4 /export/home
/home auto_home -nosuid,sec=krb4
```

Again, NFS refuses the mount if the security modes do not match.

Summary

The decisions that you make regarding the layout of file systems on a single host and, even more so, across a network will determine, to a large degree, how easy your site is to manage and how well it runs. Solaris supports a plethora of file system types, many of which are not file systems in the traditional sense, but interfaces to the system. Become familiar with the tools and options at your disposal—it will be a good investment of your time.

In this chapter, we explained that:

- File systems provide a convenient way for users to manage their data, but do not accurately reflect the way data is stored on disk.
- The Unix kernel distinguishes only a few types of files, while users think of their files as source code, compiled programs, scripts, lists, e-mail, and so on.

- Most of the information describing files (other than the contents and the file names) is stored in a file system structure called an inode.
- Logging file systems (like UFS in Solaris 7 and later) provide a degree of resistance to corruption and are, therefore, more reliable.
- Pseudo-file-systems provide access to system resources—for example, devices and running processes—by *appearing* to be regular file systems.
- Caching can be used to improve file system performance, but does not do so in every situation.
- The layout of file systems across a network can be the single most important factor with respect to performance and manageability.
- Automounter provides a tremendously useful way to automate mounting and unmounting of file systems—without requiring superuser intervention.
- NFS Logging can provide useful information on client accesses to shared file systems and directories.

We've also explained some of the directory structure of a Solaris system to make it easier for you to locate files that you might need to update.

CHAPTER 2

Planning Backups and Restores

No task in system administration has a poorer balance of work and reward than that of backing up our systems. For this reason and due to the continued pressure for increasing levels of service, new technologies for networkwide backups and high-reliability solutions that largely obviate the need for frequent backups are now beginning to take over some larger networks. For the system administrators of these sites, a trade-off is being made between tedium and complexity. We introduce some of this backup technology early in this chapter.

The rest of us still back up file systems every night and rarely, if ever, use the backups. Instead, we eject them, label them, store them, eventually reuse or replace the media on which they are written and clean the heads on the tape drives used to write them. Any script we write stands a good chance of being used a dozen times or more. Any backup we create has less than 1 chance in 100 of ever being used.

Still, the data on our systems is somehow worth the effort. Loss of a day's worth of scientific data or a script or program intrinsic to supporting operations can have an impact on our work that is way out of proportion to its seeming size and significance before the loss. For this reason, we accept the tasks of creating backups we are likely never to use and protecting them from loss and harm as one of our primary responsibilities and carry it out with boring persistence.

In this chapter, we both provide some guidance on traditional backups and introduce some technology that reduces our reliance on traditional backups by making online storage more reliable and accessible.

Networkwide Storage Solutions

While Solaris continues to support the old standbys of backup and restoration—`tar`, `ufsdump` and `ufsrestore`, and `cpio`, larger sites are rapidly moving away from the nitty-gritty tools and implementing networkwide solutions that provide not only centralized storage but also centralized backup and restoration, often of disparate systems.

With pressure on IT organizations to provide nearly round-the-clock uptime, the windows within which file systems can be unavailable are shrinking dramatically. This means that backups today have to be not only reliable, but fast. In this section, we describe the drawbacks of traditional tape backup and some of the technology that is providing network backups along with high-speed network storage.

Local Backups

Local backup strategies are prone to being outgrown. As users continue to increase the size of their file holdings and applications tend to work on larger and larger data sets, the amount of data to be backed up grows at a more or less steady pace, but the typical local tape drive has a fixed capacity that might easily be exceeded while the system is still within its useful life.

Disks are added to file servers with a much greater frequency than tape drives. Sooner or later, the tape drive is likely to be unable to accommodate the amount of data that needs to be backed up. Solutions include upgrading the backup device, going to partial or more frequent backups, and moving to RAID drives so that data is internally backed up, or *self-backing*.

However, each of these solutions has its drawbacks and ultimately adds complexity to one of the aspects of managing systems that should be as free from operational complexity as possible: consider Slee's Rule, which demands that the complexity of a task be inversely proportional to how boring it is.

In addition, even if the tape drive is able to accommodate the growing demand for backup, it is extremely unlikely to do so without an increase in downtime.

Network Backups

Backup solutions that involve backing up servers to high-capacity tape drives or tape robots might solve the problem of sizing backup capacity to meet the demand of servers with large amounts of disk space, but they will also add significant traffic to your network. Network protocols are not as efficient as operations on a local disk, especially on highly trafficked networks.

Home-brew network backup solutions, such as one might develop using `ufsbackup` with a remote drive depend on interhost trust, and can introduce security vulnerabilities if you're not careful. Commercial solutions rely on some form of remote procedure calls between client and server processes and, therefore, on the underpinnings of the related protocols.

RAID Storage

RAID devices guard against the loss of data if configured at a RAID level that provides mirroring (in which disks are duplicated) or parity (a scheme that permits some degree of lost data to be reconstructed). As we point out in Chapter 11, however, RAID offers no protection against inadvertent file corruption or deletion. If one of your users removes the source files for an important program, your backups are your only resource for recovering these files.

One of the most popular technologies for data availability, RAID is discussed in detail in Chapter 11.

Network-Attached Storage (NAS)

Network-attached storage (NAS) involves special-purpose servers dedicated to file serving. Instead of providing the general-purpose utility of a typical Unix server, these devices are optimized for file storage and retrieval and do nothing else. They typically run a Unix-like operating system that is configured solely for file serving and those tasks required to support file sharing. The NAS system will have high-performance disks and may provide a type of hierarchical storage that allows infrequently used files to migrate to slower storage.

NAS servers often support a number of network protocols so that they can be used by many types of clients. An NAS server, for example, might support AppleTalk, SMB, and NFS to facilitate sharing files among various types of clients. These servers will sometimes provide tape backup as well.

Some of the better-known NAS suppliers include Network Appliance, Auspex, and Sun Microsystems.

NOTE The acronym NAS is also sometimes used to denote a network access server—a device that provides Internet connectivity in an area, generally through dial-up connections. This unrelated term should not be confused with network-attached storage.

NAS servers often support multiple protocols—AppleTalk, SMB, and NFS—to make sharing files across platforms easier. Some NAS servers also allow for directly attached tape drives for local backups, but more often a centralized backup server is used, requiring that data be sent across the LAN. However, because backups only occur between the NAS servers and the backup server, many administrators opt to install a secondary LAN dedicated to backups. This requires the NAS server to support multiple NICs and is called *multihoming*.

Storage Area Network (SAN)

A storage area network (SAN) makes it possible to connect shared data resources to both servers and clients. Generally provided through a high-speed Fibre Channel switch, this type of storage can be made available to any number of systems simultaneously. Unlike the traditional setup in which dedicated storage is added to an individual server to increase its capacity, this solution provides "as-needed" storage that can supplement the storage on any of a number of systems.

By using a SAN, a system administrator can independently scale available storage to a particular system to demand. In addition, servers can be set up such that a single element of storage can be used by multiple servers, reducing the need to duplicate data and the overall storage requirement. Further, backup can be accomplished directly over the storage channels. This provides a considerable speed boost over using the normal LAN. Because data can be shared and because SAN systems generally provide highly reliable storage, data is more consistently available, regardless of the condition of any particular server.

The virtual storage pool can also supply storage resources to a multiplicity of operating systems. By creating this pool of storage, you are not likely to run out of storage on one system while storage is going unused on some other system simply because planning is imperfect. Instead, you can get maximum use from your "virtualized" storage volumes.

Typically, setting up a SAN will require that the system administrator establish a pool of drives and allocate portions of this pool to specific systems that do not share these devices. Reallocating devices involves removing drives from one allocation and adding them to another. Increasing the pool by adding disks provides additional drives that can be allocated on an as-needed basis. Sharing disks, however, is also possible, and a single set of disks can be shared by a group of systems. Generally, systems share disks without creating local copies of shared files, and file system software ensures that corruption is not introduced.

SAN and NAS technology, in spite of the similarity of the acronyms, represent radically different technologies. They use different protocols as well as different network devices. Most SAN technology uses Fibre Channel adapters to provide the physical connectivity between servers. These Fibre Channel adapters communicate not using NFS or TCP/IP, but using the SCSI command set. Thus, communication is more like using a local disk than communicating with a remote server. Though this idea may seem radical, consider that the SCSI protocol was designed for the reliable transfer of large amounts of data, whereas the TCP/IP protocols were designed for general-purpose communications, and NFS, in particular, includes considerable overhead for synchronizing data between clients and servers.

SAN systems also provide extremely good performance. They can support transfer rates as high as 1 gigabyte per second—and not just on short runs under the computer room floor. With fiber cabling, distances as large as 10 kilometers can be supported.

A SAN, therefore, behaves as if it were a secondary LAN connecting servers to storage. SAN-ready servers require Fibre Channel cards and Fibre Channel switches to participate in this alternate LAN. Storage arrays and tape libraries are connected to the Fibre Channel switches or hubs as well, providing reliable storage and centralized backup services without adding traffic to the office LAN.

Specialized software must be installed for managing the SAN and allocating storage pools for use on various servers and clients. If and when a system requires more storage, system administrators do not add drives on the needy system, but instead allocate additional storage on the SAN. To increase reliability, backup Fibre Channel cards can be added to the servers and clients using the SAN.

Relegating both shared storage and backups to SAN systems not only provides a high performance solution, but also offloads considerable traffic from your normal network. This in itself can provide a significant performance boost to applications across the board. In addition, backup tasks that might have previously required a number of nightly backups on any number of tape drives can be migrated to the SAN with considerable savings not only in terms of storage media but also in terms of human effort.

Just as backups can be streamlined using SAN technology, restorations can make use of more advanced technology than reading from a tape. Disks available through SAN can be provided in any form of RAID, or entire disk arrays can be mirrored. Some combination of strategies will provide you with a reliability scenario that will make data loss virtually impossible.

Though SAN technology today is based on SCSI protocols and Fibre Channel components, the high price of Fibre Channel equipment and dramatic increases in Ethernet technology may eventually push this technology back into the shadow of TCP/IP and standard networking hardware.

Backup Basics

If you're new to backups, one of the first things you need to determine is what you're going to back up and when. Your backup strategy will determine how far back you can reach to restore a file and how out of date a recovered file is going to be. If you back up every file every night, for example, you can always recover a lost file to the state it was in when the backups were run (i.e., the day before). In this section, we describe backup schedules and tools.

File Systems and Backups

As mentioned in Chapter 1, one of the factors that you should consider when laying out file systems is how and how often you will back them up. Some file systems are updated constantly and house data that is critical to your organization. The loss of data of this sort might result in lost revenue or worse. For file systems of this type, even frequent backups might not be adequate protection. Simple disk mirroring or some other level of RAID technology might be required before the risk of data loss is acceptable.

For most file systems, daily backup is generally sufficient protection against the unlikely risk of disk failure or the accidental destruction of data resulting from human error. Reproducing a day's worth of work might strain the patience of your users, but is probably within the limits of acceptable loss, especially for losses resulting from "fat fingering" or other forms of user error. Recovering yesterday's versions of files that have since been mangled or deleted will probably not set your users back too much in their work. (At least the work that your users will need to reproduce will likely still be in *their* memory banks!)

Some file systems need only be backed up on the rare occasions when they have been significantly modified. Daily backup of file systems containing only binaries and configuration files that never change is probably a waste of time as well as backup media.

TIP Keep basically static file systems separate (i.e., make sure they are separate file systems), and you will rarely have to back them up.

Housing files that change throughout the day and those that never change in the same file system is not a good strategy from the point of view of backup and restore operations. For this reason, for decades most Unix sysadmins have separated basically static file systems such as `/usr`, from those more frequently updated, such as `/home` and `/usr/local`. Keeping `/usr` "pure" (i.e., operating system files only) makes the file system stable, requiring little attention in terms of backups.

Full versus Incremental Backups

Full backups are backups in which everything in a file system is backed up. Contrasted with this, *incremental backups* include only those files that have been modified since the last backup at the same or a lower level.

Backup levels, except for level 0 (representing the full backup), are arbitrary. They mean nothing in and of themselves. Their only significance is with respect to other backups. If a backup at level 5 is made on a Tuesday night, a backup at level 6 on Wednesday would include only files that have changed since Tuesday night. If another backup at level 5 is made on Thursday night, it would also include all files that have changed since Tuesday night.

If we were to make an analogy to a family, we might say that your grandfather (level 0 backup) remembers every significant event from his own life as well as that of his son and grandson. Your father (level 5 backup), on the other hand, only remembers events from his life and yours. You (level 6 backup) only remember events from your own life. So, if Aunt Frisbee wants to be reminded of something that happened at a family reunion when your father was in grade school, she can't ask you. She needs to go back to your father or his father. Similarly, if you need to recover a file that was changed three days ago, it might not be included in last night's backup.

Limiting the backup levels you use will make your work much easier. If you only back up your files at only level 0 (full) and level 5 (incremental), for example, then every level 0 backup will contain all of your files and every level 5 backup will contain only those files that have changed since the previous backup. That's not too hard to keep straight.

TIP Don't use more than two or three backup levels. It's too much work to keep track of more than that.

Most sysadmins back up their file systems at only two or three different backup levels. Anything more complicated will probably make it too much trouble to manage the backups and keep track of which tapes cover what files for what periods of time. When you're staring at a pile of backup tapes and listening to a user desperately pleading with you to recover a file inadvertently overwritten, you'll need to determine which tape in the pile contains the most recent copy of the file before its unfortunate demise. A simple, reliable backup scheme can reduce the effort and stress at these moments.

Backup Scheduling

The traditional backup strategy calls for weekly full backups and in-between incremental backups. This strategy strikes a balance between the period for which you can still recover files and the number of backups you need to load to restore a file or a file system. With today's high-capacity tape drives, it is possible to dramatically reduce the frequency with which you need to change and label tapes. To the extent possible, backups should be automated. The corresponding restore option should be automated to the same degree.

Figure 2.1 illustrates a backup schedule in which five tapes are used to provide backups over a four-week period. As you can see, the figure represents a calendar with the day's date and the tape number in parentheses. Note how tapes 1 through 4 are each used once every four weeks, while tape 5 is used five days a week. Assume that tapes 1 through 4 (underlined) are always full backups (grandfathers), while tape 5 contains incremental backups (fathers) and is rewound at the end of each week. In this backup scenario, files can be restored from as far back as four weeks, but a file that is created and deleted in the same week (but not the current week) is no longer recoverable.

A better scheme might involve using the five tapes in such a way that each tape contains a full backup created on Saturday morning followed by a week's worth of incrementals. In this alternate scheme, any file can be recovered from the previous five weeks; the only exception would be files created and removed in a single day. Figure 2.2 depicts this alternate schedule.

If you create multiple backups (aka *dumps*) on a single tape, you might need to use the `mt -f device` command to fast forward to the desired backup in order to recover a file. The command `mt -f /dev/rmt/0n fsf 3`, for example, will jump over three backups and leave you positioned at the fourth.

Many sites, in addition to a standard weekly schedule, save one full backup each month so that it is possible to restore files as much as a year later. Whether this is a good idea for you depends on the files you use and how you use them. The cost of an extra dozen tapes may be a very small price to pay for this added protection.

TIP Back up file systems when they are not in use if at all possible. This reduces the chance that the file system will change while backups are occurring.

S	M	T	W	TH	F	SA
	1 (5)	2 (5)	3 (5)	4 (5)	5 (5)	6 (1)
7	8 (5)	9 (5)	10 (5)	11 (5)	12 (5)	13 (2)
14	15 (5)	16 (5)	17 (5)	18 (5)	19 (5)	20 (3)
21	22 (5)	23 (5)	24 (5)	25 (5)	26 (5)	27 (4)
28	29 (5)	30 (5)				

Figure 2.1 Sample backup schedule.

S	M	T	W	TH	F	SA
	1 (1)	2 (1)	3 (1)	4 (1)	5 (1)	6 (2)
7	8 (2)	9 (2)	10 (2)	11 (2)	12 (2)	13 (3)
14	15 (3)	16 (3)	17 (3)	18 (3)	19 (3)	20 (4)
21	22 (4)	23 (4)	24 (4)	25 (4)	26 (4)	27 (5)
28	29 (5)	30 (5)				

Figure 2.2 Alternate sample backup schedule

The best backup insurance is to back up all files every night, but this can be expensive insurance, not so much because of the cost of the backup media as because of the processing time of the backup operation. If backups start after midnight and are not completed until the next morning, they might start interfering with other tasks or slowing network performance. Pay attention to the time your backups take to run.

If you back up a file system every night, you can fully restore it (as needed) using a single tape. Contrast this with the possibility of requiring a full backup and three incremental backups to accomplish the same thing. Most backup strategies attempt to find a compromise between the time and expense of backups and the time and expense of recovery.

Backups versus Archives

Strictly speaking, backups and archives are not the same thing. *Backups* are routine operations intended to preserve the integrity of entire file systems from mistakes and acts of God. *Archives,* on the other hand, are snapshots of files taken at some point appropriate for their preservation—for example, when a development project is ready for beta testing or when the data for scientific analysis have been assembled in one place. Backups are generally kept for some period of time and then overwritten with newer backups. Archives are generally kept indefinitely or until the contents are no longer relevant.

TIP Encourage your users to make their own backup copies of important files. Teach them to use tar and gzip to create personal archives.

Some of the same commands are used for backups and archives, but, for the most part, file systems are backed up with `dump` commands (e.g., `ufsdump`), and archives are created with commands such as `tar` (which stands for *t*ape *ar*chive).

Backup Software

The commands most frequently used for backups include `ufsdump`, `ufsrestore`, and `tar` (these commands are discussed later in this chapter; other commands that can be used, but which are not discussed here, are `dd` and `cpio`). Most backups are initiated through `cron` and run in the middle of the night when their impact on performance is minimal and file systems are not likely to be in use. It is considered best practice for file systems to be made inaccessible when they are being backed up so that there is no chance of files changing during the backup operation, possibly corrupting the backups. For some sites, file systems must be available around the clock, and this cannot be done. Some backup software provides options for freezing file systems in ways that allow file systems to remain in use while a stable image is written to the backup media.

Commercial backup software, such as Veritas Networker and Legato Networker, sometimes provides significant advantages over home-brew solutions. The most significant of these is a front end that makes it possible for naïve users to retrieve backups of files without expert help. This feature is sometimes complemented by a choice of several versions of the same file (in case the most recently backed up version is *not* the one required). Another important advantage that commercial backup software often provides is electronic labeling and label verification. With this feature, you don't have to label tapes manually or be concerned that you might overwrite a backup you need. Some free software, such as Amanda (developed at the University of Maryland) also provides a lot of sophisticated features, such as backing up to disk if you forget to insert a tape into the drive.

Backup systems with tape robots may make it possible for you to have completely unattended backups and restores. A week or more's worth of tapes can be loaded at once and used in succession. An alternative to robot systems is a high-capacity DLT drive on which a number of backups can be written (excuse the pun) back to back. When backups are done in this way, you'll need to keep track of the number of dump images on the tape to know where to position the tape for a particular restore. The `ufsdump` utility includes a record of the file system backed up and the date and time of the backup, so it isn't difficult to examine this information and move forward and backward on the media if you find yourself using the wrong day's backup in a restore operation.

Other backup options include recently emerging Internet services that allow you to send backup data in encrypted form over the Internet to a physically remote location where your file systems are then securely stored. Although options such as these require that you trust your backups to a remote service organization, the off-site location can be a security advantage. Services such as these should be evaluated with consideration as to the level of security and the subsequent availability of data for recovery. As a security-conscious sysadmin, you will want to be confident that you, and only you, (or you and one or two trusted colleagues) can invoke a restore operation and that you will be able to gain access to your backups in a timely manner; we suggest an hour or less.

It is rare these days for systems to back up to local tape drives. Most systems, except for heavy-duty file servers, back up to remote drives. If you've followed our advice in setting up your file systems, there may be little on each client that you'd want to back up. If user's home directories, applications and data, and network services are all housed on file servers, there may be little on a client system that cannot be easily rebuilt from installation media and printouts of configuration files.

NOTE If you have important data on client systems and a central backup server, keep in mind that sending file systems over the network to back them up can seriously degrade network performance. Execute your backups during the night if at all possible.

Backup Security

Backups should be secured in a manner that reflects the sensitivity of the data they contain. It makes little sense to enforce good security practices on your network while leaving your backups unprotected. Consider how much data could walk out your door on one tape alone—and its value in the hands of a competitor or a malicious hacker.

Whether stored in a locked, restricted-access computer room or in a safe, backups should be kept out of reach of the general population of your work space. If client systems are backed up to local drives, it may be very hard to ensure that those tapes are not accessible during normal business hours and by casual visitors.

WARNING Do not consider standard office file cabinets to represent safe storage. One of the authors was taught how to pick the lock on many such file cabinets with a common X-Acto knife.

Part of the short-lived popularity of diskless clients (also called *thin clients*) was that they provided no local access to data other than via the normal login process. There were no tapes hanging out of drives or external disks to be whisked away under any circumstances.

Backup Commands

The backup commands—`ufsdump`, `ufsrestore`, and `tar`—are fairly easy to use, but the arguments vary with the media used. Tape device drivers on Solaris systems are stored in the `/dev/rmt` directory. The device `/dev/rmt/0` may represent an 8-mm drive or a DLT on your server. You'll need to know its characteristics to select the proper parameters in your backup commands.

The ufsdump Command

The `ufsdump` command is best at dumping an entire file system. As mentioned earlier in this chapter, its operation is affected by a parameter—the dump level—which determines whether the system is doing a full or incremental backup and, if incremental, how far back it should go in picking up changed files.

The format of the `ufsdump` command is:

```
ufsdump [options] [arguments] file-system
```

The options and arguments part of the command is a little unusual. Instead of using a format in which each option and argument is presented together (e.g., `-s 31120 -f /dev/rmt/0`), the command groups options together and looks for arguments to follow in the same order (e.g., `sf 31120 /dev/rmt/0`). A common `ufsdump` command might look like this:

```
ufsdump 0ubsf 126 31120 /dev/rmt/0 /home
```

In this example, the first three arguments (`126`, `31120` and `/dev/rmt/0`) align with the third, fourth, and fifth options (`bsf`). The first two options (`0` for full backup and `u` for requesting an update of the `/etc/dumpdates` file) do not require arguments. The last argument listed (`/home` in this example) is the file system being dumped.

Options for the `ufsdump` command are shown in Table 2.1.

You will often see a device with an *n* at the end of the name—for example, `/dev/rmt/0n`. This represents the *no-rewind option* of the device, which is useful, for example, if you are stacking backups one after the other on a high-capacity drive.

The `ufsdump` command can be invoked manually, but it is predominantly used in `crontab` files or backup scripts called by `cron`, as shown in the following examples:

```
11 1 * * 0-5 /usr/sbin/ufsdump 0ubsf 126 31120 /dev/rmt/0n /home
59 23 * * 0,1,2,3,4,5 /etc/daily_backup 2>&1 | mailx -s daily_backup
sysadmin
```

Table 2.1 Options for ufsdump

OPTION	ACTION
0-9	Sets the dump level
u	Updates the `/etc/dumpdates` file
b	Sets the blocking factor (default is 20 blocks per write with a 512-byte block size)
c	Indicates that a cartridge tape is being used and sets the density to 1000 bits per inch and the blocking factor to 126
d	Sets the density of the tape (default is 6,250 bits per inch)
D	Indicates that a floppy diskette is being used for the backup
f	Selects the device to write to (f stands for *file,* but this is almost always a device instead)
s	Sets the size (length) of the tape
v	Verifies content of the `ufsdump` command after completion

The following script creates full backups of some local and some remote file systems. Note that it uses the `.rhosts` facility to permit the remote dumps. If the `/.rhosts` file exists only during the backup operation, the risk is limited. Note also that it waits 2 hours before starting the dumps. It picks up the proper date for the backup when invoked at 11:59 P.M. (see the second of the two `cron` entries in the example) and then waits for a less busy time (many scripts are invoked right after midnight) to start the backups.

```
#! /bin/csh -f
#
# script to do daily backup of active partitions
set me = 'basename $0'
set rm = /bin/rm
set mt = /usr/bin/mt
set day = "'date +%a'"
set date = "'date +%Y.%m.%d'"
set bserver = "spoon"      # backup server
set device = "/dev/rmt/0n"
set echo = "/usr/bin/echo"
(date ; echo "${me}: sleeping for two (2) hours" ; sleep 7200 ; date ;
echo "")
######################################################################
##
## local - no need to use rsh
set i = spoon
ufsdump 0usf 311220 $device /home
######################################################################
##
## remote
set i = www
rsh $i ufsdump 0usf 311220 ${bserver}:${device} /WWW
######################################################################
##
## remote
set i = remhost2
rsh $i ufsdump 0usf 311220 ${bserver}:${device} /var
rsh $i ufsdump 0usf 311220 ${bserver}:${device} /etc
######################################################################
##
## cleanup
$mt -f $device rew
$mt -f $device offline
$echo "Please switch tape in DLT drive" | mailx -s "Backup Complete"
$sysadmin
```

TIP **Set up a backup user that is a member of the `sys` group. If you look at the disk devices, you will notice that `sys` has read permission. Therefore, backups do not have to run as root, yet you can still secure the backup account.**

You'll note that the backup script also rewinds and ejects the tape, and then sends an e-mail reminder to the sysadmin to put the next backup tape in the drive. There could be a problem if the backup is not completed successfully—for example, if it runs out of tape before dumping all the files. The script will hang, waiting for someone to insert another tape, the e-mail will never arrive, and it may take the sysadmin a while to notice that anything is wrong. One of the authors once ran into a case where a system being backed up to the same cartridge tape every night had been running out of tape every night for more than three months. By the time the disk crashed, the backup was practically useless, and several months of grad student data went with it. Verifying the successful completion of backups is as critical a step as initiating them.

The ufsrestore Command

The `ufsrestore` command is similar in syntax to its counterpart, `ufsdump`. Because it works with an existing dump file, however, the arguments you need to supply are reduced. Note the following command:

```
ufsrestore rvf /dev/rmt/0
```

This restores the backup from the specified device into the current directory. If you're restoring the `/home` file system, you'll want to do this:

```
spoon# cd /home
spoon# ufsrestore rvf /dev/rmt/0
```

The arguments `rvf` specify a complete restore, verbose operation (displays the names and inode numbers of the files as they are restored), and the specified tape device.

The restore operation can be run interactively, giving you the opportunity to move around in the contents of the backup and select files that you want restored. This is the easiest way to retrieve the backup copy of a single file or directory that may have been clobbered on disk. The following command initiates an interactive restore:

```
spoon# ufsrestore ivf /dev/rmt/0
```

You will see in the output of the `ufsrestore` command the date and time of the backup as well as the identity of the file system backed up. If you notice that you have the wrong tape in the drive, you can rewind and eject it and start over again with a new tape. Here's an example:

```
ufsrestore> quit
spoon# mt -f /dev/rmt/0 rew
spoon# mt -f /dev/rmt/0 offline
```

To continue with the restore, use the following commands:

```
cd                    //Changes directories
add                   //Selects files and directories to restore
```

```
ls                      //Lists files (files already selected will have
                        //their names prepended with an asterisk)
extract                 //Begins extracting the files after they have
                        //been selected
quit                    //Exits ufsrestore
```

If you are restoring select files, it is a good idea to extract them to temporary space, examine them, and then move them into place if they are the proper files and are intact. When you start up your restore operation in the `/var/tmp` directory, you will note that the extracted files will be stored in a directory structure corresponding to their original pathname appended to `/var/tmp`. For example, restoring `/home/nici/Mail/projects` in this way would result in a file named `/var/tmp/home/nici/Mail/projects`. After examining the file, you should copy it to its original location.

The tar Command

The `tar` (which stands for *t*ape *ar*chive) command is generally used to create `.tar` files from groups of related files for archival purposes, to transfer them to different locations, or to make them available to others. Most of the software archives on the Internet, for example, are in a compressed `tar` format. The file extension `.tar` is used to designate tar files, and either `tar.Z`, `.tar.gz`, or `.tgz` is used to designate compressed tar files using Unix or `gunzip` compression.

The command options for `tar` are shown in Table 2.2.

Note, for example, the following command:

```
tar cvf spectro.tar spectro
```

This backs up the directory `spectro` in the current directory to a file called `spectro.tar`.

Table 2.2 Options for tar

OPTION	ACTION
f	Specify device or archive name (otherwise refers to a specific tape device that you might not have)
c	Create archive
x	Extract from archive
v	Verbose—list files during process
t	Provide a table of contents of the archive (used to verify contents after the archive is created or before extracting from it)

The tar command works from the directory in which it is invoked and incorporates the directory structure from that point of view in the archives it creates. For example, the archive created with the preceding command would contain files with the following structure:

```
spectro/1999-09/nw
spectro/1999-09/sw
```

If this archive were extracted in the directory `/var/tmp`, it would create these files:

```
/var/tmp/spectro/1999-09/nw
/var/tmp/spectro/1999-09/sw
```

This is important to know when extracting files from an archive. If the archive was created from the root file system, for example, the embedded files will contain pathnames starting with a slash (/), and `ufsrestore` will attempt to restore the files to the full pathname specified. This might mean that they will end up in your root file system. If this isn't what you intended and you don't have room for the files there, this could present you with a problem. One workaround is to create a symbolic link that will move the target directory somewhere else.

The `tar tvf` command will list the files included in a tar file. This is useful in determining what the resultant file names will be and in determining whether the archive includes what you are looking for without having to extract it.

Summary

Backups are a vital part of system administration. Though seldom used, they are nevertheless important when they are needed. The smart sysadmin will minimize the work involved by automating the backup tasks and preparing a restore script that facilitates the retrieval of files.

Even if backups run attended, you should make some effort to ensure that they are running smoothly and being completed successfully. If your backup script sends e-mail to you at the completion of each backup, you can be confident that your file systems are backed up. If you fail to notice when this e-mail does not arrive, however, you may be lulling yourself into a false sense of security.

The best advice that we can offer about backups is this:

- Keep the manual effort to a minimum.
- Be as familiar with the process of restoring files as you are with the process of backing them up. If you back up your file systems with a script, have a script for restoring them, as well.
- Make sure someone else knows the routine. If you change tapes every night, someone else must know how and when this is done.

- If it is at all possible for users to retrieve their own files from backup (obviously without affecting the proper functioning of the service or introducing security risks), encourage this. All other things being equal, it is generally reasonable to provide the most motivated individuals with the means for accomplishing a task.

It's also a good idea to test your backups once in a while to ensure that you can recover files easily in an emergency. Even the simple process of fast forwarding over a series of backups on a single tape can be painful if you haven't done it in a couple of years.

CHAPTER

3

Booting and Hardware Diagnostics

With Solaris, the process of starting with cold hardware and winding up with a system ready for users to log in is flexible and elegant. Understanding how this process works when all is well will help you know what to do on the rare occasions when you have trouble. There are numerous options at boot time that involve not only the boot process but also hardware settings and diagnostics. The tests that you can run before booting will help you understand and fix problems. Commands available to you at boot time also provide a lot of information about a system's layout—the attached devices, the PROM version, and so on—this is useful in making decisions about installations and upgrades.

The boot process comprises the processes that start up between the time you turn on the power or use the `boot` command to the time that the system is ready for logins. These processes include everything from core operating system services to application software that is specific to your site or to a single system.

This chapter describes commands available at the OpenBoot prompt. It explains the boot process, the `boot` command, configuration parameters, and troubleshooting. It also provides commands for setting up an alternate boot disk.

The PROM Monitor and OpenBoot Firmware

The firmware in a Sun system's *programmable read-only memory* (PROM) is executed whenever you turn a system on. After a short sequence of hardware tests, the system's PROM loads the bootblock from the boot disk. The *bootblock* is a fixed-size chunk of data (i.e., a block) that contains the location of the secondary booter. This, in turn, then loads the secondary bootblock (created during system installation or with the `installboot` command). At this point, you have barely withdrawn your fingers from the system's power button. The secondary bootblock then loads a standalone program into the system's memory and starts running it. When booting from a local disk, the standalone program is `ufsboot`. When booting off a network, `inetboot` is loaded.

The standalone program loads the system kernel and starts the `init` process, which then takes over control of the booting process. `init` first reads the `/etc/default/init` file to set up environment variables and determine which run state to assume. Figure 3.1 illustrates the boot process up to this point. The remainder of the boot process is described in Chapter 4.

If you interrupt a system during normal booting (i.e., by entering the Stop-A or L1-A sequence), you will find yourself communicating with one of two programs. The most basic of these, which offers you a > prompt, is a very limited monitor program that listens to the keyboard and offers you the command choices shown here:

```
Type b (boot), c (continue), or n (new command mode)
>
```

You can restart the boot at this point by entering **b**. If you enter **n** (which, as you can see from the preceding output, stands for *new command mode*), you'll see the prompt change to ok, telling you that you are now interacting with a more sophisticated Forth program instead (see the nearby sidebar if you don't know what Forth is). The ok prompt is the Forth language's normal prompt. The particular Forth program that you are running, called the *OpenBoot* program, does several things. It supplies hardware diagnostics, provides you with access to a number of parameters that influence the boot process as well as the operations of the system when fully booted, and allows you to boot the system from one of a number of devices. In the OpenBoot environment you can, for example, specify a different disk as the default boot disk or modify the amount of memory that is tested on boot-up.

Though new Sun users may think that the only thing they can type at the ok prompt is **boot** or **boot cdrom**, this is far from true. Forth is a small but complete programming language, and it is sitting there under your fingertips. There are now three major versions of the OpenBoot firmware that you are likely to encounter, depending on the particular Sun SPARC-based hardware you are using—version 2.*x* (e.g., 2.2), version 3.*x* (e.g., 3.1), and version 4.*x*. Each version offers a collection of commands that you can use to interact with your hardware. The exception is the Sun Fire midrange servers, which use OpenBoot 5.*x* firmware written in Java and C.

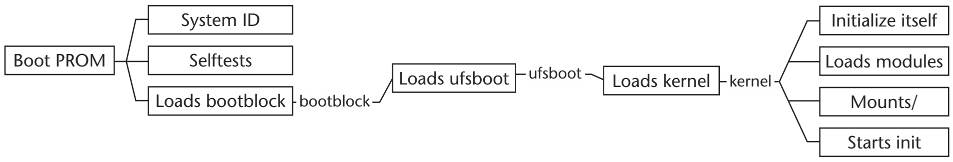


Figure 3.1 The boot process.

KNOWING FORTH . . .

If you've never heard of the Forth language, it is an interpreted language that was popular for a while among companies developing embedded systems because it is so compact that the controlling software can easily be resident on a single chip. Unlike Java and the many Java-related languages, Forth has been around for a long time. For applications where the entire computer may be on a single board inside some small device—for example, on a spacecraft or in your car—this language could have much to offer.

For you, while booting your Sun system, Forth offers a way to start the process that, in turn, starts all the other processes associated with an up-and-running Unix environment. In fact, this single strategy works across a family of systems (with quite diverse hardware) through the loading of the appropriate device drivers into the *nonvolatile random access memory* (NVRAM).

In fact, there is a *device tree* available to you through the OpenBoot firmware. The device tree corresponds, in a less than completely obvious way, to the file system tree that resides under the `/devices` directory of a running system. You can move around within this tree, probe devices or list their attributes, and run diagnostic tests.

If you want to convince yourself that you're actually interacting with an interpreted language, try giving it some commands to execute. We have to warn you, however, that Forth is a language that uses *postfix* (i.e., operator last) notation. If you want it to add two numbers, you have to put the + sign at the end. You should type, for example, **1 2 +** to add the numbers 1 and 2. Forth will give you the answer and then offer another ok prompt. To define a simple procedure, try the following (typing in the bold entries). These commands create, run, and then display the code for a procedure that multiplies a number by 10. Postfix is used on the first line for multiplication and the second with a period (.) that removes the computed value from the stack and displays the following result:

```
ok : times10 ( n1 ) 10 * ;
ok 3 times10 .
30
ok see times10
: times10
   10 *
;
```

Most of the commands shown in Table 3.1 are available to you at the ok prompt. This list is not exhaustive and some of these commands may only exist in specific versions of the OpenBoot firmware (i.e., this is dependent on your hardware). Refer to the release notes for your particular hardware for a complete listing of available commands.

Table 3.1 OpenBoot Commands

COMMAND	ACTION
`.attributes`	Displays attributes for the device node (2.*x*)
`.enet-addr`	Displays the current Ethernet address
`.fregisters`	Displays values in registers %f0 through %f31
`.idprom`	Displays ID PROM contents, formatted
`.locals`	Displays values in the input, local, and output registers
`.properties`	Displays attributes for the device node (3.x)
`.registers`	Displays values in registers %g0 through %g7, plus %pc, %npc, %psr, %y, %wim, and %tbr
`.traps`	Displays a list of processor-dependent trap types
`.version`	Displays version and date of the boot PROM
`banner`	Displays power-on banner
`boot`	Boots the system
`cd`	Changes the directory in the device tree (2.*x*)
`cd ..`	Changes directory to the parent node
`cd/`	Chooses the root node of the device tree (2.*x*)
`dev`	Changes the directory in the device tree (3.*x*)
`dev..`	Chooses the device node that is the parent of the current node (3.x)
`dev/`	Chooses the root node of the device tree (3.x)
`devalias`	Displays the real devices associated with a device aliases (such as disk0). With arguments, `devalias` defines aliases for devices (e.g., `devalias alias path`)
`firmware-version`	Shows version of OpenProm firmware (e.g.,0x00030001 represents 3.1)
`go`	Resumes a program which was interrupted with the L1-A or Stop-A sequence)
`help`	Lists categories of help available
`help category`	Lists help available under a given category; for example, `help diag` lists help available under the `diag` category

Table 3.1 *(Continued)*

COMMAND	ACTION
`ls`	Lists the nodes under the current nodes
`nvalias alias device-path`	Stores a device alias in NVRAM
`nvunalias alias`	Deletes the named alias from NVRAMRC
`password`	Sets a PROM password that will be required for boot
`printenv`	Displays all current configuration parameter values
`printenv boot-device`	Displays value of the default boot device
`printenv bootfile`	Displays value of the default boot file
`printenv parameter`	Displays value of the specified parameter
`probe-scsi`	Identifies SCSI devices attached to only the built-in SCSI bus
`probe-scsi-all`	Identifies all SCSI devices attached to the system
`pwd`	Displays the pathname of the current node
`reset`	Resets the system, similar to a power cycle
`set-default parameter`	Resets the specified parameter to factory default
`set-defaults`	Resets all parameters to factory default
`setenv parameter value`	Sets the named parameter to the given value
`show-bus`	Displays all the installed and probed Sbus devices
`show-devs`	Displays the entire device tree
`sync`	Synchronizes all mounted file systems and flushes out all dirty pages to the disks before reboot
`test disk`	Executes the named device's method selftest, if available
`test-all`	Tests all the devices in the system
`unsetenv parameter`	Unsets the named parameter
`watch-clock`	Tests the clock function
`watch-net`	Monitors the network traffic; displays a dot (.) each time it receives an error-free packet
`words`	Displays the values for the current node

The Boot Command

The easiest way to boot a system from the powered-down state is, of course, to turn it on. When you turn a system on, the system locates the boot device using information stored in its NVRAM and starts loading various boot software, as described earlier. If you interrupt this process with a Stop-A or your `auto-boot?` NVRAM parameter (discussed later in this chapter) is set to `false`, you can choose to boot from a different device or explore the OpenBoot environment.

The full syntax of the `boot` command is as follows:

```
boot device-specifier filename options
```

Here *device-specifier* is one of a number of devices that can supply the boot software, *filename* is the name of the program to be booted (e.g., stand/diag), and *options* can be either -a, so that you are prompted for the name of boot file; -h, which will halt the boot process after the program specified as filename has been loaded; or -r, which rebuilds device drivers on boot-up (needed if you add a disk or other SCSI device to the system). You can also specify the run state that you want to assume, if you want to assume a run state other than the default one.

To boot from a device other than the default one (boot disk), there must be a bootblock available on the device in question. If, for example, you want to boot from disk1, which corresponds to `/sbus/esp/sd@1,0` (the default disk is `/sbus/esp/sd@3,0`), you must have at some point used the `installboot` command to create a bootblock on this disk (see the "Setting up an Alternate Boot Disk" section later in this chapter). To boot off the network, you must have prepared a boot server to accommodate this system as a boot client.

NOTE The drive designators that are used at this point resemble the Solaris 1 naming conventions more than those used in Solaris 2. Regardless, they refer to the same devices as those you'll see on a booted system as `/dev/dsk/c0t1d0s0`, and so on.

You can alternately boot from other devices (if they are set up for this), including (but not limited to) those listed in Table 3.2.

These labels (e.g., cdrom) are not the actual addresses of the devices, but are device *aliases* that make it easier to refer to any of these devices. The defined device aliases will differ from one system to the next. You can obtain a complete listing of the device aliases on a system with the OpenBoot firmware's `devalias` command. A list of typical aliases is displayed in Table 3.3. The device aliases provide the location of the device drivers.

To establish a new alias, use the `nvalias` command (`nvalias alias device-path`). For example, if you want to refer to your alternate boot disk as `altdisk`, you can create that alias with the following command, provided you have a bootblock installed on this disk:

```
nvalias altdisk /sbus/esp/sd@2,0
```

Table 3.2 Alternate Boot Devices

DEVICE ALIAS	CONDITION
cdrom	You must have the installation CD in your drive.
disk2 (the second disk)	The disk must contain a bootblock.
floppy	You must have a boot floppy disk in your drive.
net (Ethernet)	You must have a boot server prepared to support the local system.
tape	You must have a boot tape in your drive.

The `nvunalias` command deletes an alias.

TIP A useful convention might be to create an `altdisk` alias on all systems that have an alternate boot disk available. Then, regardless of which of the attached disks is serving as the alternate boot disk on a particular system, you can use the same command for booting from it.

Table 3.3 Typical Device Aliases

ALIAS	BOOT PATH	DESCRIPTION
`disk`	`/sbus/esp/sd@3,0`	Default disk
`disk0`	`/sbus/esp/sd@3,0`	Same as above
`disk1`	`/sbus/esp/sd@1,0`	Second internal disk
`disk2`	`/sbus/esp/sd@2,0`	First external disk
`disk3`	`/sbus/esp/sd@0,0`	Second external disk
`floppy`	`/fd`	3.5” diskette
`tape`	`/sbus/esp/st@4,0`	First tape drive
`tape0`	`/sbus/esp/st@5,0`	Second tape drive
`cdrom`	`/sbus/esp/sd@6,0:c`	CD-ROM, partition c
`cdroma`	`/sbus/esp/sd@6,0:a`	CD-ROM, partition a
`net`	`/sbus/le`	Ethernet

Boot Options

The familiar automatic booting on power-up is, surprisingly, not mandatory. Even this setting can be modified within the OpenBoot environment. If you set the `auto-boot?` parameter to false, the system will not boot when you power it on, but will instead, wait at the > or the ok prompt for you to enter the next command. This can be a useful feature if you plan to run diagnostics before you fully boot a system and don't want to have to enter a Stop-A sequence at the right moment to interrupt a normal boot.

You can change the boot disk as well as the boot file that is normally used in the boot process. If you have an alternate boot disk and need to boot from it rather than the default one (which may have been damaged), you can specify this information at the ok prompt. To change the disk that is used for booting, use the `setenv boot-device` command. To change which file is used as the boot file, use the `setenv boot-file` command. The configuration parameters that control the boot process are described in Table 3.4.

TIP If only a few of your systems have alternate boot disks set up, it is a good idea to label them with instructions for its use—for example, "Alternate Boot Disk. To use, type boot disk2 at the ok prompt."

Table 3.4 Boot Configuration Parameters

PARAMETER	DESCRIPTION
`auto-boot?`	Controls whether or not the system will automatically boot after a system reset or when the power is turned on. This parameter is normally set to `true`.
`boot-command`	Specifies the command to be executed when `auto-boot?` is set to `true`. The default is `boot`.
`diag-switch?`	If set to `true`, this parameter causes the system to run in Diagnostic mode. This is normally set to `false`.
`boot-device`	Contains the name of the default boot device to be used when OpenBoot is not in Diagnostic mode.
`boot-file`	Contains the default boot arguments to be used when OpenBoot is not in Diagnostic mode.
`diag-device`	Contains the name of the default Diagnostic mode boot device.
`diag-file`	Contains the default Diagnostic mode boot arguments.

Table 3.5 Boot-Time Abort Commands

COMMAND	DESCRIPTION
Stop-A	Aborts boot
Stop-D	Enters diagnostics mode
Stop-F	Enters Forth (OpenBoot) environment
Stop-N	Resets NVRAM contents to default values

The Stop-A (same as L1-A) sequence is a fairly drastic operation if run on an operational system. Proper shutdown of a system with the `init 0` command is *far* preferable. On boot-up, however, before file systems have been mounted, there is little, if any, consequence to running this command. Alternatives to the Stop-A sequence available in the later revisions of the OpenBoot firmware are listed along with Stop-A in Table 3.5. Using the Stop-F sequence is the same as doing a Stop-A; it aborts the boot process and leaves you at the ok prompt. The Stop-N sequence resets all the factory defaults, similarly to using the `set-defaults` command at the ok prompt. The Stop-D sequence aborts the boot and leaves your system in Diagnostic mode. Diagnostic mode loads a special diagnostic program that allows you to run a collection of hardware tests. System administrators who have been working on Sun systems a while are probably familiar with the `boot diag` command and the interface provided for running tests ranging from monitor testing to port checks.

Configuration Parameters

One of the commands you will use often with the OpenBoot firmware is `printenv`. In similar manner to the same-named `/bin/csh` command, `printenv` displays the names and values of parameters. As far as the OpenBoot firmware is concerned, these are the hardware configuration parameters that reflect the settings of your system hardware and influence the boot process. The `printenv` command displays the values of all current parameter settings, along with the default values. Sample output from this command follows:

```
Parameter Name              Value          Default Value
Selftest-#megs              1
oem logo                    5a 5a 5a 5a    5a 5a 5a 5a
oem-logo?                   false          false
oem-banner
oem-banner?                 false          false
output-device               screen         screen
input-device                keyboard       keyboard
sbus-probe-list             0123           0123
keyboard-click?             false          false
```

```
keymap
ttyb-rts-dtr-off              false          false
ttyb-ignore-cd                true           true
ttya-rts-dtr-off              false          false
ttya-ignore-cd                true           true
ttyb-mode                     9600,8,n,1,-   9600,8,n,1,-
ttya-mode                     9600,8,n,1,-   9600,8,n,1,-
diag-file                     vmunix
diag-device                   net            net
boot-file                     vmunix
boot-device                   disk           disk
auto-boot?                    true           true
watchdog-reboot               false          false
fcode-debug                   false          false
local-mac-address?            false          false
use-nvramrc?                  false          false
nvramrc
screen-#columns               80             80
screen-#rows                  34             34
sunmon-compat?                true           true\
security-mode                 none           none
security-password
security-#badlogins           0
scsi-initiator-id             7              7
hardware-version              xxxxx
last-hardware-update
testarea                      85             0
mfg-switch?                   false          false
diag-switch?                  false          false
```

You can change defaults, including, for example, where the output displays when you're booting. This can be very useful information if a monitor fails and all you have on hand to replace it is a dumb terminal. In this case, you will want to send system output as well as system input to the terminal, but the only thing you will have at your disposal is your system keyboard. If you blindly issue the following commands on power-up (assuming that the terminal is attached to serial port A and reset), you should have a working terminal from which to continue your diagnostics and booting:

```
Stop-A
n
setenv output-device ttya
setenv input-device ttya
```

You can reset all of the parameters to their default (factory) settings with the `set-defaults` command or a single parameter to its default with the command `set-default parameter`. You can change the value of a parameter with the command `setenv parameter value`. However, most parameter changes do not take effect until the next time you power-cycle the system or use the reset command (which is virtually the same as a power cycle). Table 3.6 lists the parameter display and setting commands.

Table 3.6 Parameter Display and Setting Commands

COMMAND	DESCRIPTION
`printenv`	Displays current configuration parameters
`setenv parameter value`	Sets specified parameter to value provided
`set-default parameter`	Resets specified parameter to the factory default
`set-defaults`	Resets all parameters to the factory defaults

When changing parameters, keep in mind that the OpenBoot firmware will not check what you enter to ensure that it is a proper setting for your hardware. An incorrect setting may keep your system from booting. Resetting all of the parameters to the default or determining the correct setting for the parameter entered in error will be necessary before such a system will boot on its own.

Perusing the Device Tree

The set of commands that you use to move around the device tree (corresponding to your system hardware) and examine features and attributes varies somewhat between the two versions of the OpenBoot firmware. Table 3.7 provides the commands and descriptions.

Table 3.7 Device Tree Navigational Commands

COMMAND	DESCRIPTION
`.attributes`	Displays the names and values of the current node's properties (2.*x*)
`.properties`	Displays the names and values of the current node's properties (3.*x*)
`cd node-name`	Finds a node with the specified name, starting with current location node in the tree (2.*x*)
`cd..`	Selects the node that is parent to the current node (2.*x*)
`dev node-name`	Finds a node with the specified name, starting with current location node in the tree (3.*x*)
`dev..`	Selects the node that is parent to the current node (3.*x*)
`device-end`	Deselects the current device and exits the tree
`ls`	Lists the names of nodes that are children of the current node
`pwd`	Displays the path name that corresponds to the current node
`show-devs [path]`	Displays devices directly under the specified device in the device tree
`words`	Displays the currents node's methods

As mentioned earlier, the device tree corresponds to the devices on your system and provides you with the means of running tests on these devices when the system is not yet booted. This low-level access to your hardware may be the only way that you can isolate certain problems.

Depending on the particular OpenBoot firmware revision, the command that you will use to move around the device tree and run tests will be slightly different. In any case, we strongly advise that you become comfortable with working at this level sometime when you have spare time and no emergencies. Later on, when an important system is not booting and you have to use these commands to determine what is wrong, you will be much more comfortable.

TIP **Plan for disaster. Reserve a system that you can play with. Practice alternate boots. Run diagnostics just for fun. When a real disaster strikes, you'll be ready for it.**

Similarly to moving around a Unix file system, navigating within the device tree uses simple commands. For OpenBoot 2.*x* systems, the command is the familiar `cd`. For the later version, 3.*x*, the command is `dev`. In either case, you can think of the device structure that you are working within as if it were a treelike file system. Instead of directories, this structure has *nodes.* Nodes correspond to the structure of hardware devices.

To view the entire device tree, use the `show-devs` command, as shown here:

```
ok show-devs
/options
/virtual-memory@0,0
/memory@0,0
/sbus@1,f8000000
/auxiliary-io@1,f7400003
/interrupt-enable@1,f4000000
/counter-timer@1,f3000000
/eeprom@1,f2000000
/audio@1,f7201000
/fd@1,f7200000
/zs@1,f0000000
/zs@1,f1000000
/openprom
/packages
/sbus@1,f8000000/bwtwo@3,0
/sbus@1,f8000000/le@0,c00000
/sbus@1,f8000000/esp@0,800000
/sbus@1,f8000000/dma@0,400000
/sbus@1,f8000000/esp@0,800000/st
/sbus@1,f8000000/esp@0,800000/sd
/packages/ibx?-tftp
deblocker
disk-label
```

The listing you will see depends on your particular hardware, of course, but you will see something roughly resembling what we have shown here. Each line in this output represents a device. If you compare this listing to what you see under `/devices` on your up-and-running system, you will recognize many of the devices:

```
sunstation:/devices 9# ls
audio@1,f7201000:sound,audio     sbus@1,f8000000
audio@1,f7201000:sound,audioctl  zs@1,f1000000:a
eeprom@1,f2000000:eeprom         zs@1,f1000000:a,cu
profile:profile                  zs@1,f1000000:b
pseudo                           zs@1,f1000000:b,cu
sunstation:/devices/sbus@1,f8000000/esp@0,8000000 36# ls -a
.             sd@3,0:f      sd@6,0:d      st@4,0:bn     st@4,0:ln
..            sd@3,0:f,raw  sd@6,0:d,raw  st@4,0:c      st@4,0:m
sd@3,0:a      sd@3,0:g      sd@6,0:e      st@4,0:cb     st@4,0:mb
sd@3,0:a,raw  sd@3,0:g,raw  sd@6,0:e,raw  st@4,0:cbn    st@4,0:mbn
sd@3,0:b      sd@3,0:h      sd@6,0:f      st@4,0:cn     st@4,0:mn
sd@3,0:b,raw  sd@3,0:h,raw  sd@6,0:f,raw  st@4,0:h      st@4,0:n
sd@3,0:c      sd@6,0:a      sd@6,0:g      st@4,0:hb     st@4,0:u
sd@3,0:c,raw  sd@6,0:a,raw  sd@6,0:g,raw  st@4,0:hbn    st@4,0:ub
sd@3,0:d      sd@6,0:b      sd@6,0:h      st@4,0:hn     st@4,0:ubn
sd@3,0:d,raw  sd@6,0:b,raw  sd@6,0:h,raw  st@4,0:l      st@4,0:un
sd@3,0:e      sd@6,0:c      st@4,0:       st@4,0:lb
sd@3,0:e,raw  sd@6,0:c,raw  st@4,0:b      st@4,0:lbn
```

When located at a node corresponding to a particular device, you can determine what, if any, self-tests are available for that device and run them.

Booting over the Network

Though the typical Solaris boot may involve no more than turning the power on or entering **boot** at the ok prompt, network booting is also possible as part of the procedure for installing a system or as its normal method of booting. A client that acquires its address using RARP (the Reverse Address Resolution Protocol) provides its Ethernet (i.e., MAC) address and requests its IP address in return. In this case, the syntax of the `boot` command is:

```
boot <path-to-network-device>
```

On the typical system, the device alias `net` will already be defined to be the path to the network device so that `boot net` will be an equivalent command. To boot using DHCP, on the other hand, the command or the `net` alias must contain the word DHCP (e.g., `/pci@1f,0/network@c,1:dhcp`).

The client during a network boot can supply no more than its Ethernet address or it can supply up to eight additional pieces of information, including:

- dhcp or bootp for address discovery
- The boot server's IP address

- The name of the boot program to be loaded by tftp
- The client's IP address
- The IP of the router
- The number of boot retries before declaring that the boot has failed
- The number of tftp retries before declaring the tftp operation to have failed
- The subnet mask for the local network

The arguments must be presented in the order given, with commas inserted to cover for missing parameters. The full syntax of the boot command is:

```
boot network-device:[dhcp|bootp,][server-ip],[bootfilename],
[client-IP], [router-IP],[boot-retries],[tftp-retries],
[subnet-mask][boot-arguments]
```

Troubleshooting Your System

If you're having trouble booting your system or aren't sure that the attached devices are working properly, you can take advantage of the diagnostic utilities available through the OpenBoot firmware. After all, there are occasions when this will be all that you have to work with. We have used the `probe-scsi` command, explained in the next paragraph, many times to determine whether an attached disk that appears to be nonfunctional on a booted system might actually be working. If a `probe-scsi` does not respond with an entry for a device that is attached and turned on, you can be confident that the disk is fundamentally hosed. Most likely, there is a problem with the controller hardware on the disk itself (but please be sure that you're using a known-to-be-good SCSI cable!).

The `probe-scsi` command provides information on each device attached to the built-in SCSI bus. If you have additional SCSI buses, use the `probe-scsi-all` command instead. Both commands will probe the attached devices and display the information that they retrieve. This usually includes information on the manufacturer and model number of each device. Importantly, the SCSI target address is also included. If you were to inadvertently press the SCSI selector of an external device while moving it, you might notice this fact when examining the `probe-scsi` output. Once a SCSI target no longer matches what has been defined in the system, your system will not be able to communicate with it. If you have many external devices, the likelihood of creating a SCSI target conflict is higher. The `probe-scsi` command will help you make this kind of determination without requiring you to climb over your desk and read the SCSI selector while hanging upside-down.

To test an installed device, use the `test device-specifier` command. This command will execute the self-test defined for the particular device. Here are some examples:

```
ok test floppy
Testing floppy disk system. A formatted
disk should be in the drive.
```

```
Test succeeded.
ok
```

The preceding command tests the floppy drive on the system.

```
ok test /memory
Testing 64 megs of memory at addr 4000000 11
ok
```

This command generally tests a portion of memory.

```
ok test net
Internal Loopback test - (result)
External Loopback test - (result)
ok
```

This command tests the network loopbacks.
Other commands include:

```
ok eject-floppy
```

This command ejects the diskette in the drive.

```
ok watch-clock
Watching the 'seconds' register of the real time clock chip.
It should be ticking once a second.
Type any key to stop.
12
ok
```

This command allows you to determine whether the clock timing is correct. The time between increments of the counter should be 1 second.

```
ok watch-net
Internal Loopback test - succeeded
External Loopback test - succeeded
Looking for Ethernet packets.
'.' is a good packet. 'X' is a bad packet.
Type any key to stop
....................................
ok
```

You can use the `test-all` command to test all of the devices. Tests will be run only for devices that have defined self-tests, of course, but this is more the rule than the exception.

To test the system clock, use the command `watch-clock`. The response should be a tick every second until you stop the command. If you have reason to suspect problems with your system clock (e.g., complaints from `cron` that it is having to make adjustments), this command will help you pinpoint the problem.

Table 3.8 Commands that Display System Information

COMMAND	PURPOSE
`banner`	Displays the power-on banner
`show-sbus`	Displays a list of installed and probed sbus devices (compare with `probe-scsi`)
`.enet-addr`	Displays the Ethernet hardware address associated with the network interface
`.idprom`	Displays the contents of your ID PROM
`.traps`	Lists trap types
`.version`	Shows the version and date associated with the boot PROM

The `watch-net` command both tests your Ethernet loopback connection and tells you whether packets are arriving on the Ethernet interface. If you do not see a string of dots, you are not properly connected to your network or you have a problem with your network interface. You might want to check the cable and/or transceiver that connects to the system or the port on the hub to which this cable connects. If you see *X* characters interspersed with a series of dots, your interface is receiving bad (probably malformed) packets. This probably indicates a problem elsewhere on your network.

A series of commands provides you with the ability to display information about your system (see Table 3.8).

Setting Up an Alternate Boot Disk

A corrupt root file system can keep a system from booting. If you'd like a quick workaround for critical systems, install a second root file system and make it bootable. This solution is most effective on systems that have a reasonably sized root file system—in other words, *not* one of those root file systems where everything is in root.

The ideal candidate for an alternate root is a separate drive or an unused partition on a different drive than your current root. An unused partition on the same drive is okay, but less desirable because it might be affected by whatever evil is going to befall your original root file system.

Once you've identified your partition (let's say you've salvaged a half-megabyte drive from an old client), attach it to your system and reboot (see Chapter 1). Then make sure it is formatted and prepared with a file system of the approximate size of the one that it will be backing up. You might have to partition the drive. If so, refer to Chapter 1 if you need help.

Mount the alternate root partition on `/mnt` (the universal mount point). You don't need or want to create an entry in your `/etc/vfstab` file because you won't be using it in any capacity during normal operations.

The next thing you want to do is copy your existing root file system to the new one. The easiest thing to do is to copy the entire file system. If your `/var` directory is in root, you may want to go through the trouble of emptying out log files; they will be outdated by the time you need to use the alternate partition, so there's little sense in preserving them. Commands such as `cat /dev/null > /mnt/var/adm/wtmp` will make this process fairly painless (but don't forget to include `/mnt` in the pathname!).

There are numerous ways to copy the root partition. Don't use the `cp` command! To create a duplicate of a file system, the best approach is to use a back-to-back `tar` command or a back-to-back dump/restore, as shown in the following examples:

```
spoon# cd /; tar cpBf - | (cd /mnt;tar x -)
spoon# ufsdump 0f - / | (cd /mnt; ufsrestore xf -)
```

Both of these commands preserve ownership and permissions of all the copied files—vital to the functioning of a root file system. Once the copy operation has completed, you need to modify the `/etc/vfstab` in the alternate root so that it refers to itself as the root file system. Look for a line that looks like this and change the first two arguments to reflect the new partition:

```
/dev/dsk/c0t3d0s0   /dev/rdsk/c0t3d0s0   /   ufs    1   no   -
```

Unmount `/mnt` and label the disk (assuming it's an external drive) with the commands needed to boot from it in any emergency. Depending on your system, the address of the alternate root will be different from what we show here. Verify this on your system with the `probe-scsi` command explained earlier in this chapter:

```
ok devalias disk2 /sbus/esp/sd@1,0
ok boot disk2
```

The only tricky part to this process is finding and installing the right bootblock. The following commands should do the trick. Replace the device name with your device's. The command `uname -i` will put you in the correct directory for your system:

```
spoon# cd /usr/platform/`uname -I` /lib/fs/ufs
spoon# installboot bootblk /dev/rdsk/c0t1d0s0
```

For Solaris *x*86 systems, the installboot syntax is slightly different; there's an extra bootblock to install. Use this command instead:

```
spoon# installboot pboot bootblock /dev/rdsk/c0t1d0s0
```

Your alternate root partition is now ready to come to your rescue.

Summary

A wealth of information about the configuration and health of your systems is available behind the modest little ok prompt. The diagnostics and boot options available to you might come in extremely handy when a hardware problem arises. For example, if a new disk does not respond when a system is booted or a network interface appears inoperable, you can use `probe-scsi` or `watch-net` to gather critical information about these components.

Key things to remember during the normal boot process include:

- System administrators frequently add or modify the start/stop scripts in the `/etc/rc?.d` directories that control which processes start and stop in each of the Solaris run states.
- Careful setup and testing of these scripts is essential.
- We also provided instructions for creating an alternate, bootable, root partition. The steps involve creating an alternate root file system and then making it bootable.

CHAPTER

4

Configuring Run States

Every Unix system has processes that must start "automagically" when the system is turned on or rebooted. Most of these configure and initialize basic operating services including mounting file systems, starting network communications, monitoring the keyboard for login activity, listening for email or print requests, and invoking many other processes that provide the eventual users with the services they require.

The process through which all of these services spring to life somewhere between a cold boot and a working system is elegantly designed and is both easy to understand and easy to modify. Given a basic understanding of how the important files are organized and invoked, you can provide support for additional services, ensuring that they are started and stopped as required in the life cycle of your Solaris systems.

The first and most important thing to recognize—even before you `cd` over to `/etc/init.d` or start vi to create a new start/stop script—is that the configuration of run states "out of the box" reflects a careful orchestration of system resources. The prudent system administrator, or *sysadmin,* as we affectionately call him or her (and as all three of us have proudly called ourselves for many years), will change little or nothing of this default setup. What he or she will do, however, is add scripts for site-specific processes and use the scripts already in place to start and stop processes as needed on running systems.

Picturing Run States

Without overdoing the case for the value of images in understanding concepts related to operating systems, allow us to give you our image of what a run state represents. Most references seem to define run states by enumerating the run states that are possible and then associating each with a brief description—for example: run level 3, which is multiuser mode with NFS support. Though we appreciate these descriptions, we prefer to stress that each run state is effectively the collection of processes that start and stop as that run state is invoked. Please keep this in mind as you consider the run state descriptions in Table 4.1.

In *run state 0,* basically nothing is running; the system is halted. Only the monitor process or the OpenBoot environment is available. In *run state 1* or *s,* single user, limited processes are running. Run state 1 is limited because the processes that support login through devices other than the console, log system messages, run `cron` jobs, mount remote file systems, and provide the bulk of the usual Unix environment are simply not running. In single-user mode, you will, obviously, have some local file systems mounted and be able to log in as root on the system console. Naming support and network information services, such as DNS and NIS, will not be available. This run state is used for system repair and troubleshooting (e.g., checking file systems). In *run state 2,* on the other hand, most processes are running, but file systems are not shared (exported). Users with homes on the system will find they cannot access them.

There is nothing peculiar about run states. They have nothing to do with the state of the CPU or even the kernel. The only real difference between one run state and another is, once again, what processes are running. It is not until we get to run state 2 that we begin to see the familiar environment we have come to identify as Unix.

Configuring run states is simply a matter of selecting which processes are to start and stop when that run state is entered and then setting up the proper scripts to make this happen. More precisely, it is a matter of selecting which processes *stop* and *start.* We will explain this distinction shortly.

Table 4.1 Run State Descriptions

RUN STATE	DESCRIPTION
0	Halted
1, s	Single user
2	Multiuser, no NFS
3	Multiuser with NFS
4	Not used
5	Interactive reboot
6	Reboot

The init Process

You will recall from Chapter 2 that control during the boot process eventually rests with the init process. With process ID (PID) 1 and owner root, init is the single process that can be said to be parent to all other processes running on a Unix system—in any of its run states (other than halted, of course). This is why experienced (read: "been there, made that mistake") system administrators fear the mistyped `kill -1 1` command—meant to be `kill -HUP 1` (`kill 1 1` will ungraciously halt the system). When the init process gains control early in the boot process, the first thing it does is read its configuration file. This file, /etc/inittab, is one that every senior sysadmin should know well. Though few, if any, of us will ever modify this file, it is so critical to the boot process that familiarity with its format and function is part of the essential knowledge that none of us can afford to be without. Here is what it looks like:

```
ap::sysinit:/sbin/autopush -f /etc/iu.ap
fs::sysinit:/sbin/rcS     >/dev/console 2>&1 </dev/console
is:3:initdefault:
p3:s1234:powerfail:/usr/sbin/shutdown -y -i5 -g0 >/dev/console 2>&1
s0:0:wait:/sbin/rc0        >/dev/console 2>&1 </dev/console
s1:1:wait:/usr/sbin/shutdown -y -iS -g0  >/dev/console 2>&1
</dev/console
s2:23:wait:/sbin/rc2       >/dev/console 2>&1 </dev/console
s3:3:wait:/sbin/rc3        >/dev/console 2>&1 </dev/console
s5:5:wait:/sbin/rc5>/dev/console 2>&1 </dev/console
s6:6:wait:/sbin/rc6        >/dev/console 2>&1 </dev/console
fw:0:wait:/sbin/uadmin 2 0 >/dev/console 2>&1 </dev/console
of:5:wait:/sbin/uadmin 2 6 >/dev/console 2>&1 </dev/console
rb:6:wait:/sbin/uadmin 2 1 >/dev/console 2>&1 </dev/console
sc:234:respawn:/usr/lib/saf/sac -t 300
co:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` console
login:" -T sun -d /dev/console -1 console -m ldeterm,ttcompat
```

The first thing you're likely to notice as you look at this file is that it is columnar, with fields separated by colons. Look at the third line; the third field in this line reads initdefault. This is the run state that the system will assume when you boot, unless you specify a different run state at the ok prompt. The other lines in /etc/inittab have the following format:

```
id:rstate:action:process
```

The first field, id, is simply a label identifying the entry. The second identifies the run state in which the particular entry will be used. The third details how the process will be run; we'll talk about this in a moment. The fourth specifies the process that will be run.

Although there are numerous options that can be specified in the third field—respawn, wait, once, boot, bootwait, powerfail, powerwait, off, ondemand, and initdefault—only half are used in the out-of-the-box /etc/inittab. Let's look at these:

```
spoon% cat /etc/inittab | awk -F: '{print $3}' | uniq
sysinit
initdefault
powerfail
wait
respawn
```

Of these, the most common is `wait` (i.e., wait for the process to complete), and the next most common (other than `sysinit`) is `respawn`. You can read the man page on inittab for an explanation of each of the action fields, or refer to Table 4.2, but the function of the `respawn` keyword is especially worth noting, so we'll discuss it here.

The `respawn` action is basically a `keep alive` directive. If any process that is started by init with this keyword specified in the action field dies, `init` will restart it. If you, as superuser, kill either the service access controller (`/usr/lib/saf/sac`) or the port monitor (`/usr/lib/saf/ttymon`), `init` will restart it. Clearly, very few processes on a system warrant this kind of electronic babysitting. However, these two services are critical to your access to the machine, so they must be started up again if they go down for any reason.

Table 4.2 Actions in /etc/inittab

ACTION	MEANING
`boot`	Executed only when `/etc/inittab` is read for the first time
`bootwait`	Executed when changing states from single user to multiuser; `init` waits for completion of process before moving to the next appropriate entry
`initdefault`	Specifies the run state that `init` will assume if no other state is explicitly requested
`off`	Process to be stopped when entering the specified run state
`once`	Starts specified process without waiting for completion; executed once only
`powerfail`	Specifies process to be executed if `init` receives a `powerfail` signal; read and remembered for later use
`powerwait`	Works like the `powerfail` signal, except that `init` waits for the process to be completed
`respawn`	Starts the process on entering the run state and starts again if process dies at any point
`sysinit`	Run before logins are accepted
`wait`	Runs process and waits for completion

Now, let's pick up again with the process of the normal system boot. The init process reads the initdefault line to determine which run state to enter. If the field is blank, init prompts the user for this information. If the field has more than one digit, it sets the target run state to the highest of those present. It executes the sysinit entries (the first two lines in the inittab file) sometime before you see the login prompt on the console.

In a normal boot, init is then going to run the process specified in each line that includes a 3 in the rstate field (i.e., the lines labeled s2 and s3). Discounting the respawn and powerfail entries (these are special cases), this leaves us with these entries:

```
/sbin/rc2                              >/dev/console 2>&1 </dev/console
/sbin/rc3                              >/dev/console 2>&1 </dev/console
```

Of these, /sbin/rc2 is the primary run script for run state 2, and /sbin/rc3 is the same for run state 3. These two scripts, therefore, drive the boot process, initiating all the processes that you are used to seeing in an up-and-running Solaris system. The way that they do this is quite clever and dictates the steps that you must take in amending the boot process with your own site-specific processing. This process is illustrated in Figure 4.1.

The information in the /etc/inittab file is used whenever a system's run state is changed, not only on initial booting. For example, if you issue the command init 2 on a system in run state 3, init will start run state 2. It will use the s2 entry and then run the /sbin/rc2 script. This script will, in turn, execute the kill scripts in /etc/rc2.d, stopping the NFS file service, but will not execute the start scripts (because we are coming down a run state, the processes are already running and a line in the /sbin/rc2 script prevents this from happening). Subsequent calls to init specifying the *same* run level are ignored.

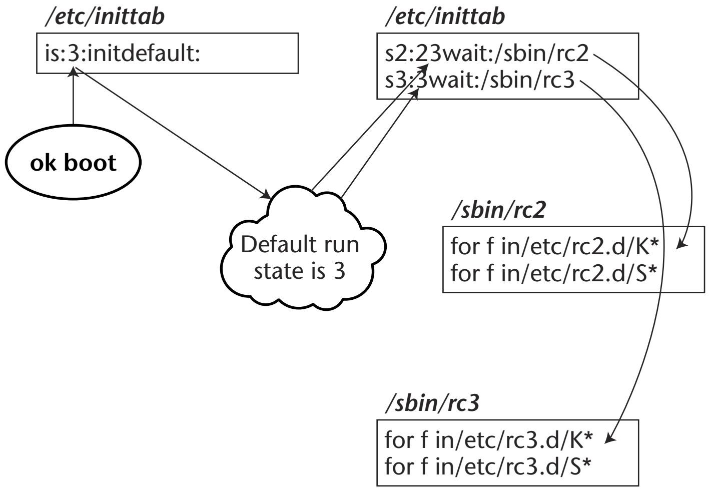


Figure 4.1 The normal boot sequence.

The rc Scripts

If you look at either of these two scripts (`/sbin/rc2` and `/sbin/rc3`), you'll note that they do several things. First, they look at certain parameters related to the state of the system and make some basic processing decisions based on their value. More specifically, note that each of the scripts—`rc1`, `rc2`, and `rc3`—uses the `who -r` command. You may be familiar with this command as the way to tell what run state your system is in. If you run this command on most systems, you'll get a response like this:

```
unixhost% who -r
    .       run-level 3  May  1 08:14     3      0  S
```

Though the man page tells us only that this command indicates the run level, it clearly does more than this, and the `rc` scripts make use of the additional information provided. When the `rc2` script determines whether `x$9` is equal to `xS`, it is simply evaluating the last field in the response to the `who -r` command. The set part of the `set '/usr/bin/who -r'` command assigns each field in this response to a separate variable, as shown here:

```
congo% who -r
    .       run-level 3  Nov 30 19:08     3      0  S
    ^           ^     ^   ^  ^    ^       ^      ^  ^
    |           |     |   |  |    |       |      |  |
    |           |     |   |  |    |       |      |  |
   $1          $2    $3  $4 $5   $6      $7     $8 $9
```

`$9` is the ninth field in this response (in this example, `S`), so `x$9` is `xS`. The `x` is a scripting trick to ensure that there will be some text for the comparison regardless of the response of this command. In other words, we will not wind up with invalid syntax.

These lines in the run script are essentially saying, "If previous run state was `S` (i.e., single user), then issue the message "'The system is coming up. Please wait.'"

```
if [ x$9 = "xS" -o x$9 = "x1" ]
then
     echo 'The system is coming up.  Please wait.'
     BOOT=yes
or if the 7th was 2, then this:
elif [ x$7 = "x2" ]
then
        echo 'Changing to state 2.'
```

The bulk of the work of the `rc` scripts is done in two for loops, the order of which is well worth noting. First, these scripts run through what we shall call the *kill scripts* in the associated directory—`/etc/rc2.d` for rc2 and `/etc/rc3.d` for rc3. The kill scripts are all the files in this directory starting with a capital letter *K*, picked out by the following for command:

```
for f in /etc/rc2.d/K*
```

Then, the `rc` scripts run what we shall call the *start scripts,* again in the associated directory. The start scripts are all the files in this directory that start with a capital letter *S,* as identified by the following line of code (executed only if you are coming from some state other than 2 or 3):

```
for f in /etc/rc2.d/S*
```

TIP Read through one of the rc scripts—preferably `/sbin/rc2` or `/sbin/rc3`—sometime when you're bored. Then compare the contents of `/etc/init.d` with `/etc/rc2.d`. This will give you a useful view of how run states are effected.

Kill and Start Scripts

Start and kill scripts respectively start up and stop individual processes or groups of related processes in the order determined by their names (e.g., `K02*` would run before `K17*`).

It's important to note that the K and S scripts are actually the same files with different names. In fact, they are set up as *hard links*. Recall from Chapter 1 that hard links provide a way of giving a single file more than one *identity* in the file system. The overhead associated with hard links is very small. They only use the space required by the directory entry. They share inodes and disk space with the original files. If you modify one of a set of hard links, you modify them all, since they really are nothing more than multiple references to a single file.

TIP Always use hard links when creating start/stop scripts. Put the original in `/etc/init.d` and the link in the appropriate `/etc/rc?.d` (usually `/etc/rc2.d`) directory. Because hard links share a common inode, this practice will allow you to more easily relate a set of scripts for a particular service.

As an aside, a number of Unix commands are actually hard links as well. Consider the `mv` and `cp` commands. They are actually the same file. The way to tell that this is the case is to do a long listing including the inode number, as shown here:

```
congo% ls -li mv
39720 -r-xr-xr-x   3 bin     bin       15992 Oct 25  1995 mv
congo% ls -li cp
39720 -r-xr-xr-x   3 bin     bin       15992 Oct 25  1995 cp
```

There is a *correct* way to implement start/stop scripts. In short, you want to follow the model that the system uses. Build each script in the `/etc/init.d` directory, giving it a meaningful name that relates to the service it starts and stops, (e.g., `/etc/init.d/apachectl`). Include processing for both a start and a stop argument if it's appropriate. You might include a restart option and a usage statement as well.

Then, after carefully selecting the points at which these processes should be started and stopped, add the `K*` and `S*` hard links to the `rc?.d` directories, as shown in this example:

```
ln /etc/init.d/apachectl /etc/rc3.d/S98apachectl
ln /etc/init.d/apachectl /etc/rc2.d/K56apachectl
```

These lines indicate that we want to start the `httpd` process late in the sequence of processes that start when we enter run state 3. Further, we want to kill the process when we enter run state 2.

TIP Test your start/stop scripts by manually invoking them (as root) sometime when no one's work will be disrupted.

The kill and start scripts are not only used during boot-up and with changes in the run state (as invoked with the init command). Most sysadmins who have made more than one Big Mistake will also use the scripts in the `/etc/init.d` directory to start and stop processes whenever the need arises. After all, these scripts are generally coded with considerable care. If more than one process needs to start, or a process ID file (which identifies the process ID of a process) needs to be created or removed, the scripts will include the code to make this happen. Shutting down or restarting processes with these scripts is, therefore, the *safest* way to do so. For example, a restart of an Apache daemon might look like this:

```
www# /etc/init.d/apachectl restart
```

A service without a restart option might be restarted like this:

```
www# cd /etc/init.d; mysql stop; mysql start
```

Further, if you are adding your own kill and start scripts, you should follow the coding conventions that the existing scripts use. Create parameters, enclose commands in if statements that ensure that a file exists before you try to invoke it, and use a case statement to handle start and stop parameters. The following script is a skeleton to be used for creating start/stop scripts that actually do something.

```
#!/bin/sh                                # Identify the shell.
start=/usr/local/bin/start               # Create parameters as needed.
stop=/usr/local/bin/stop
case "$1" in                             # Test the argument.
'start')                                 # Set up start and stop
options.
        if [ -f $start ]; then
                echo "starting your service";
                    $start
        fi
        ;;
'stop')
```

```
        if [ -f $stop ]; then
                echo "stopping your service";
                    $stop
        fi
        ;;
*)
        echo "Usage: $0 { start | stop }" # Offer usage statement
        ;;
esac
exit 0
```

TIP A good way to disable a script temporarily is to change its first letter from a capital to a small letter. Scripts starting with lowercase s or lowercase k will be ignored by the `/sbin/rc?` scripts, but will otherwise retain complete knowledge of what they were set up to accomplish. In other words, they can easily be put back into service when the need arises. If you use the `mv` command, they will also retain their hard-link relationship to their counterpart in the `/etc/init.d` directory.

Good work habits with respect to start and stop scripts will simplify managing system services. When your scripts are hard links, for example, you can easily identify a set of related scripts by searching by inode using a `find` command such as this:

```
cd etc; find . -inum 6803 -print
```

WARNING The kill/start scripts are Bourne shell scripts. Some force Bourne shell execution by specifying `#!/bin/sh` (sometimes referred to as the "shebang" line) on the first line. Others do not. Be careful if you change the root's default shell or if you execute any of these scripts from a different shell environment. Better yet, insert the shebang line into any scripts that lack it and, while you're at it, ensure that root's umask is 077 or 027 (or add `umask 077` or `umask 027` to the scripts).

Summary

The configuration of run states on Solaris systems is elegant and easy to manage—at least, it is once you understand how the file structure is intended to work. Here are the most important points to remember.

- Some processes will be restarted by init any time they die. Even the superuser cannot effectively kill these processes.

- When you boot a system with the default `inittab` configuration, it will first run the `/sbin/rc2` script and then `/sbin/rc3`. Most processes are started through `/sbin/rc2`.
- You can change run states at any time with the `init` command. The `init 0` command is the proper way to shut down a system, but you should warn users—local users and those who may be using shared file systems.
- You can change the default run state if you have good reason.
- You can add your own scripts to start and stop application processes during boot-up and shutdown operations. Proper startup and shutdown operations are critical to the operation of certain applications.
- Start/stop scripts should all have the same basic format. Follow the coding guidelines in this chapter to keep your scripts from aborting and to allow their use during boot-up or normal operations.

CHAPTER

5

Installing and Patching Your Solaris System

This chapter provides information on installing or upgrading your Solaris systems and applying patches to maintain the integrity of your systems.

The installation method choices have changed dramatically in recent releases of Solaris. Solaris 9 systems, for example, can be installed with any of five separate methods. Depending on how many systems you manage and the similarity of these systems, one method may save you considerably in terms of time and trouble.

Preparing for Installation

Most new Sun systems will arrive preinstalled and ready to be configured for the roles they will play on your network. The only thing that keeps these systems from being ready for your network is the information that makes them a part of your network (e.g., hostname, IP address, default router) and some basic security information (e.g., a root password). If the default configuration of these systems meets your needs, the only thing you have to do is turn them on and answer a series of configuration questions. In other words, these systems are ready for what is called a *Factory JumpStart* installation.

Older systems will be reinstalled or upgraded. A system being reinstalled will be rebuilt from scratch. Any information on the disks will be overwritten. A system being upgraded will retain configuration information.

This section shows you how to prepare for a Solaris installation.

Gathering Necessary Information

Before any kind of Solaris installation, you should have a general idea how the system is to be used so that you can lay out file systems and select software appropriate to this role.

Knowing What Information You Need

Here's a checklist of things we think you should know installing or upgrading your system:

- The hostname to be used
- The domain name
- The IP address of the system
- The default router
- The name service to be used
- The name and IP address of the NIS, NIS+, LDAP, or DNS server
- The subnet mask
- Remote file systems that are to be mounted
- Any unusual requirements for disk usage
- Whether you want to configure power management
- Whether any additional languages should be installed
- Whether you will be using DHCP or IPv6
- The root password to be assigned
- The security policy (e.g., whether you need to install Kerberos)
- The time zone
- The role the system is to serve (server, client)
- The appropriate software cluster (e.g., end user, developer)
- If the system is to be a server, the number of clients and/or users to be served

Things to Keep in Mind When Gathering Information

Here are a few helpful hints that we have when you gather information about your system:

- Don't overlook the fact that most of this information will be readily accessible on a system *before* it is upgraded.
- Don't fail to double-check the host's name and address information.
- Maintain logs with the vital statistics of all of the systems you'll be managing, including who the primary user is.
- Installation time may be the only time you touch some systems in a year or more. Use it as an opportunity to update your inventory, as well.

- During installation, update a prepared form with all pertinent information about the system, including the primary user, the names of anyone with root access, the location of the system, the peripherals, and identifying information.

Calculating Space Requirements

The recommended space for Solaris 9 is considerably larger than that for preceding versions of the OS. Table 5.1 below provides recommendations for each of the primary clusters (also referred to as *software groups*).

These recommendations include space for file system growth. Even so, you probably won't be trying to install Solaris 9 on a system with only 2 GB of hard disk.

What to Expect from an Installation

After you collect the information that you're going to need for an installation, the rest of the process is remarkably straightforward. The authors recall the days when it was necessary to know what disk controller was resident on your system in order to install the operating system. Solaris installation has become considerably easier since then. The forms presented to you during installation are straightforward and the process is, on the whole, painless.

The installer will walk you gently through the steps involved in the installation process. You will spend a short period of time answering questions and a long period of time while the files are read off the installation CD or DVD, or over the network. Ideally, you would start an installation before leaving for the day and check the next morning to be sure the installation has completed properly.

NOTE Because Solaris 9 is packaged on a set of two CDs (or one DVD), you will not want to wait while the system reads files during the installation. Be prepared for one or more long periods of time in which there is nothing for you to do—especially if you are installing an older system with a slow CD-ROM drive.

Table 5.1 Disk Space Requirements for Solaris 9

CLUSTER	RECOMMENDED DISK SPACE
Entire Software Group + OEM (includes some additional third-party software)	2.9 GB
Entire Software Group	2.7 GB
Developer Software Group	2.4 GB
End-User Software Group	2.0 GB

New file systems are created as needed (e.g., if you change partitioning or are doing a full installation) and then populated. The process of installing a system from scratch can take many hours to complete. Most of this does not involve any interaction with you. For this reason, we generally start the process and then go off to other tasks (or home for the day).

If you make any wrong choices during installation, don't panic. You can fix almost anything without going through the process again. You can add packages you omitted and change answers you provided. In Chapter 1, we show you where the answers to various installation questions are stored so that you can find and fix any installation-time configuration mistakes—for example, in the hostname or domain name, DNS server, IP address, and so on.

Things to Keep in Mind If You're Upgrading a System

If you are installing Solaris on a system running an earlier release, you may have the option of doing an upgrade instead of a full installation. An upgrade will only overwrite files that have changed since the last release and will leave most, if not all, of your configuration files intact. Check your release notes, as the option to upgrade will be dependent on your system type and your current configuration.

Anytime you are going to reinstall or upgrade a system, back it up first. You will probably never need anything off these backups, but if you do, it will be a simple matter to fetch a configuration file you don't want to recreate from memory or notes. Installation-time disasters are rare indeed, but you don't want to be the first in your company to have a system you can't return to its prior state.

Installation Methods

In this section, we examine the various installation methods and describe their pros and cons. Solaris 9 provides these methods of installation:

- Web Start
- suninstall
- JumpStart
- Solaris Live Upgrade
- Factory JumpStart

In addition, Solaris 9 introduces *Web Start Flash*, a tool for creating a reference configuration that can be used to clone a master system. If you're going to be installing a lot of systems of the same basic configuration, Web Start Flash may speed up each installation considerably. A Web Start Flash archive can be used with any of the other installation methods.

If you have many systems of the same type, JumpStart is still the method of choice. The investment you make up front—in understanding the steps involved and setting up the necessary files—will pay off quickly and repeatedly. In Chapter 6, we tell you how you can get the most advantage from JumpStart. If you have only a few systems to install or if your systems are all different, the investment you would have to make to create Web Start Flash archives and JumpStart profiles would probably not pay off sufficiently to make the effort worthwhile.

Preconfiguring System Information

All Solaris installation methods require that information about the system's local (i.e., network) identity and environment (e.g., default router) be provided so that the system in question can be configured properly for the network on which it is going to participate. Several installation methods, notably Web Start, suninstall, and custom JumpStart, can detect and use configuration information that is prepared ahead of time and stored in a special file called the `sysidcfg` file. When you use one of these files with any of these installation methods, you will not be prompted to supply this information during the installation. Much of this information can also be configured within a name service. We will cover only the `sysidcfg` file in this chapter.

A `sysidcfg` file must be prepared for each system that is to be installed. Because it contains the bulk of the information that will be different from one system to the next, regardless of how much they are otherwise alike, this makes sense. However, preparing a single file and then modifying it for each system is not difficult.

The keywords that can be used in a `sysidcfg` file are listed and explained in Table 5.2.

Table 5.2 Keywords for Use in sysidcfg Files

KEYWORD	DESCRIPTION	EXAMPLE
`name_service`	Name service, domain name, and name server-NIS, NIS+, DNS, LDAP, and NONE have different options	`name_service=DNS {domain_name=foo.com name_server=192.1.9.2 search foo.com}`
`network_interface`	Network interface, hostname, IP address, netmask, and DHCP and/or IPv6	`network_interface =hme0 {hostname= dragonfly default _route=10.1.1.9 ip_address=10.4.3.111 netmask=255.255.255.0 protocol_ipv6=no}`
`root_password`	The "encrypted" password for root (see /etc/shadow)	`root_password =b0c0ffee54a$e`
`security_policy`	Security policy for the system	`security_policy =kerberos {default _realm=foo.com admin_server=xxx.foo. com kdc=kdc.foo.com, kdc2.foo.com}`

(continues)

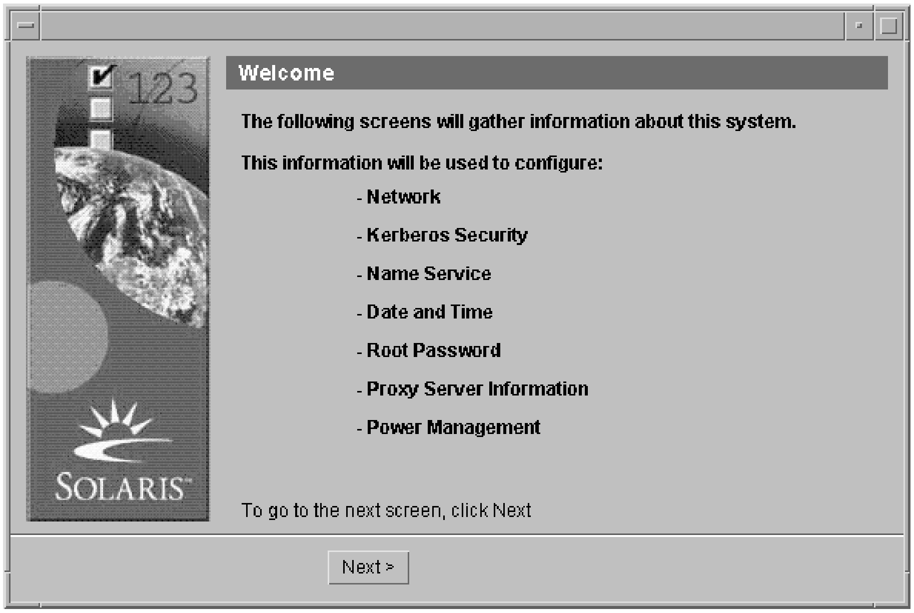
Table 5.2 Keywords for Use in sysidcfg Files *(Continued)*

KEYWORD	DESCRIPTION	EXAMPLE
`system_locale`	Language for install and desktop	`system_locale=en _US.ISO8859-1`
`terminal_type`	Terminal type	`terminal=vt100`
`timezone`	Time zone	`timezone=US/Eastern`
`timeserver`	Date and time (time server or *localhost*)	`timezone=localhost`

After being configured, `sysidcfg` files can be made available to clients through a shared NFS file system (use `add_install_client` with the -p option) or by copying them to the base of the file system on a UFS floppy.

Using Web Start

Chances are, if you are installing Solaris on an individual system that is currently running an older version of Solaris, you will install using Web Start. Requiring no particular preparation (other than collecting the information on the system and doing your just-in-case backups), Web Start walks you through the installation while providing access to information about the new release. The Web Start interface (i.e., the Welcome screen) is shown in Figure 5.1. You must have a local CD or DVD drive to use Web Start.



The following screens will gather information about this system.

This information will be used to configure:

- Network
- Kerberos Security
- Name Service
- Date and Time
- Root Password
- Proxy Server Information
- Power Management

To go to the next screen, click Next

Figure 5.1 Web Start Welcome screen.

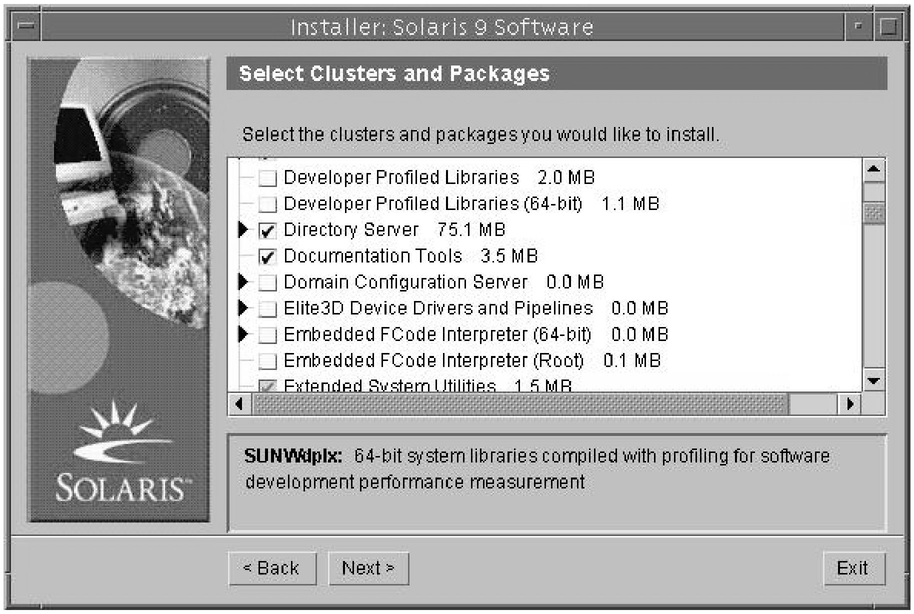
Web Start provides both a GUI and a command-line interface (referred to generically in Sun documentation as a "CLI"). One of the advantages of Web Start is that it provides an easy way to move back and forth during the installation process. If you realize that you've made a mistake or have to go back and make a change before the software is loaded onto your system, you can simply use the Back buttons to back up as far as you need to go.

If the software detects that the system has a video adaptor, it will automatically use the GUI. Notice the Back button in Figure 5.2. You can also exit the installation procedure at any time before the software begins to load by clicking on the Exit button on the bottom right of the window.

Solaris 9 is packaged both as a set of CDs and on a DVD. Either medium can be used for the installation, depending on the type of drive in your system. Obviously, you cannot use the DVD if the system being upgraded has only a normal CD device. The installation media also contains an electronic copy of the installation manuals. During the installation process, you can refer to any information on the release—from a "what's new" type of overview to fairly detailed information on the installation process. Links are available in the upper-left corner of the screen.

The Web Start software will determine whether the system can be upgraded. Upgrades are considerably faster. However, the dramatic change in disk space requirements may make it impossible for you to install a system without realigning file systems.

Web Start loads software into swap space in the process of upgrading the system. If you are using the CD set, you start with the Solaris installation CD. Halt the system, insert the CD into the CD drive, and enter **boot cdrom** at the ok prompt. If you have a DVD, insert it into the DVD drive and enter **boot cdrom** just as you would with a CD drive.



Select the clusters and packages you would like to install.

- ☐ Developer Profiled Libraries 2.0 MB
- ☐ Developer Profiled Libraries (64-bit) 1.1 MB
- ✓ Directory Server 75.1 MB
- ✓ Documentation Tools 3.5 MB
- ☐ Domain Configuration Server 0.0 MB
- ☐ Elite3D Device Drivers and Pipelines 0.0 MB
- ☐ Embedded FCode Interpreter (64-bit) 0.0 MB
- ☐ Embedded FCode Interpreter (Root) 0.1 MB
- ✓ Extended System Utilities 1.5 MB

SUNWdplx: 64-bit system libraries compiled with profiling for software development performance measurement

Figure 5.2 Deselecting a software component.

The system will prompt you as each of the additional CDs is required.

NOTE If you are using the DVD, you will have everything needed for the installation on the single DVD.

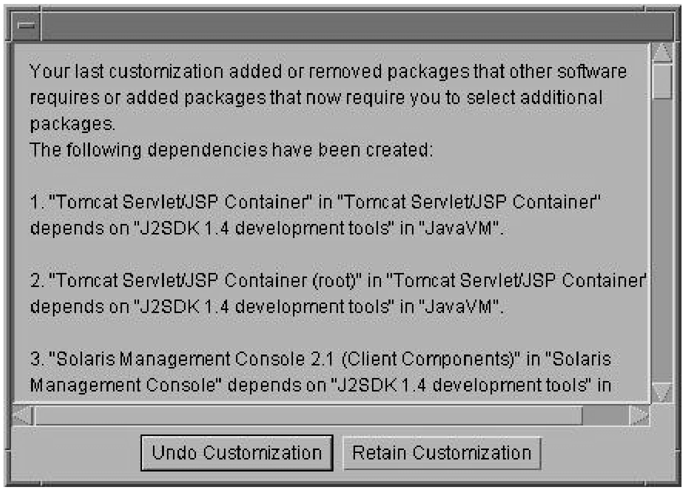
Selecting a software cluster does not prevent you from adding or removing software components from the selected cluster. You can customize the software to be loaded during the installation. Web Start will inform you of dependencies if you attempt to remove a component that is required by another component or if you attempt to add a component that relies on components not yet selected. If a dependency is detected, you can elect to resolve or ignore it. Figures 5.2 and 5.3 illustrate this process.

After the installation is complete, you can view information about the process that was collected in log files stored at these locations:

```
/var/sadm/system/logs
/var/sadm/install/logs
```

You will also be notified if the installation fails.

Web Start is a good choice for an installation method anytime you are installing a single system or a number of systems that are different in their basic configuration (e.g., different models, amount of disk space). However, a Web Start installation can take as long as several hours on a system with a slow CD drive or on a congested network and cannot compete with JumpStart if you are installing dozens or hundreds of systems of the same type.



Your last customization added or removed packages that other software requires or added packages that now require you to select additional packages.
The following dependencies have been created:

1. "Tomcat Servlet/JSP Container" in "Tomcat Servlet/JSP Container" depends on "J2SDK 1.4 development tools" in "JavaVM".

2. "Tomcat Servlet/JSP Container (root)" in "Tomcat Servlet/JSP Container" depends on "J2SDK 1.4 development tools" in "JavaVM".

3. "Solaris Management Console 2.1 (Client Components)" in "Solaris Management Console" depends on "J2SDK 1.4 development tools" in

Figure 5.3 Web Start detecting a software dependency.

Using suninstall

The suninstall tool provides only a command-line interface and is the oldest of the installation methods available today. In its standard form, suninstall requires you to supply configuration information in response to a series of prompts. As we shall show in a moment, however, a special file can be prepared with configuration information and then be used in lieu of this extended exchange of data.

Solaris suninstall can be found on the Solaris 9 CD #1. It does not provide any means of installing software in addition to the Solaris operating system. You will need both CDs to use suninstall and also the language CD if you want to load optional languages. You can also perform a suninstall installation if the software is loaded onto an installation server.

To initiate suninstall from a local CD, insert the Solaris CD #1 into the drive and enter **boot cdrom** at the ok prompt.

If you preconfigured system information using a `sysidcfg` file, the suninstall tool will be able to proceed without requiring this information piecemeal from the installer. Otherwise, suninstall will request each piece of information as it is needed.

WARNING Sun warns that some local configuration modifications might not survive a suninstall upgrade.

Using JumpStart

JumpStart is the method of choice for installing many systems of the same type. It provides the means for the installer to prepare a description of the configuration of these systems and use it repeatedly to install any number of systems in the same way. Based on *profiles*, JumpStart allows you to create installation rules based on the characteristics of systems that you are installing. In addition, you can specify scripts that are run prior to and following the installation of the operating system. You can use these scripts to further customize the system or to install additional software.

For example, one of the authors uses JumpStart to install servers that are used as test boxes for the software he develops. Each time a new product is to be tested, a server is rebuilt using JumpStart and a standard profile. The finish script then installs additional software that is needed, such as Oracle, Apache, iPlanet, and JRun, and configures these applications for the particular system. By using JumpStart, the configuration of each test system at the beginning of testing adheres to an accepted standard and little needs to be done to ensure that the systems are in a known state when testing begins.

NOTE JumpStart is not a good choice if you are installing only a few systems or if your systems have little in common with each other. The resources required to configure and maintain a JumpStart server precludes this choice for such sites.

Chapter 6 deals exclusively with JumpStart, so we will say no more about it here.

Using Web Start Flash

The Web Start Flash tool enables you to create a *reference installation* or *archive* that contains all files on the system being cloned. This archive can subsequently be used to install any number of systems that are, not surprisingly, called *clones*. In other words, you create a system that reflects the way you want any number of other systems to be configured and then capture this configuration by creating a Web Start Flash archive from that system. This archive is then used in the installation process to more quickly set up virtually identical systems.

NOTE Selecting a Web Start Flash installation method means that you will be fully installing (not upgrading) a system. It also implies that each system built with this tool will be an exact replica (with respect to configuration) of the master.

To use Web Start Flash, you need to do all three of the following things:

- Install and customize the master system from which the archive will be created (once)
- Create the Web Start Flash archive (once)
- Use the Web Start Flash archive to install a system (any number of times)

Web Start Flash is described in detail in the next chapter.

Performing a Solaris Live Upgrade

The live upgrade option for installing a Solaris system makes it possible for you to upgrade a system with the least amount of downtime. The technique involves setting up a duplicate boot environment while the system continues running off of the original. You then upgrade the duplicate environment or install from a Web Start Flash archive. The original environment remains unaffected.

To be eligible for a live upgrade, a system must be running Solaris 2.6, 7, or 8. These systems can be live-upgraded to Solaris 8 or Solaris 9. You need to have enough spare disk space to duplicate critical file systems—those file systems that are to be modified by the upgrade. In addition, you need to have certain packages installed, including the software that comprises the live upgrade tool.

The live upgrade operation involves several steps:

- Copying file systems to be upgraded to new locations
- Upgrading the copied file systems
- Making the new set of file systems active and rebooting the system

File systems can be merged or separated in the copying process. If you have `/`, `/usr` and `/opt` in separate partitions and want them to be in a single partition when you activate the new environment, you can do this. Similarly, if they exist in a single file system and you want to separate them, this is also possible. The steps required for a Solaris live upgrade are shown in Figure 5.4.

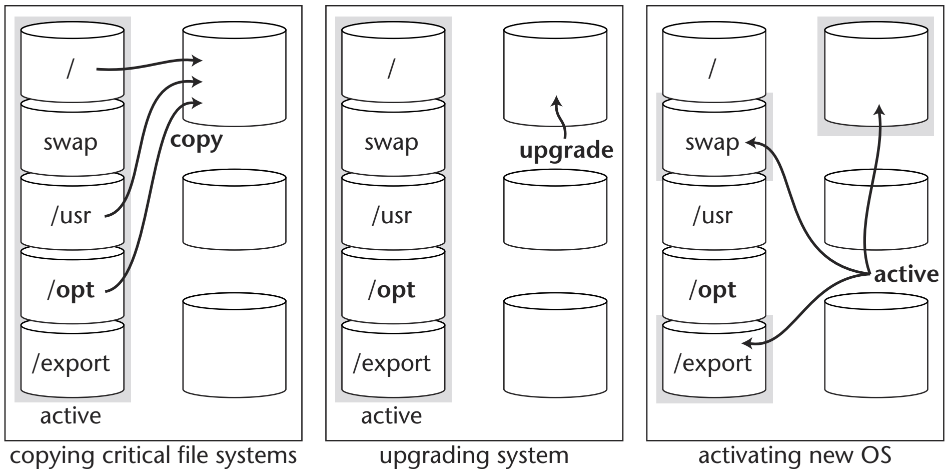


Figure 5.4 Solaris Live Upgrade.

Partitions for the new file systems must be as large as the recommended sizes for the new release. The slice chosen for the new root must be a slice from which the system can boot and cannot be a Veritas VxVM volume. It can be on a different disk or on the same disk. Figure 5.4 is not meant to imply that you must have a separate disk for the new installation.

Packages required for the Solaris Live Upgrade are listed in Table 5.3.

You can check whether these packages are installed by using the `pkginfo` command as follows:

```
pkginfo | egrep "SUNWadmap|SUNWadmc|SUNWlibc"
```

Table 5.3 Solaris Live Upgrade Packages

SOLARIS 2.6	SOLARIS 7	SOLARIS 8
SUNWadmap	SUNWadmap	SUNWadmap
SUNWadmfw	SUNWadmc	SUNWadmc
SUNWadmc	SUNWlibc	SUNWlibc
SUNWmfrun		SUNWbzip
SUNWloc		
SUNWlibc		

NOTE Make sure that you use the Solaris Live Upgrade software from the installation software of the *target* system. If you are upgrading a system to Solaris 9, for example, install the software from the Solaris 9 DVD or CD #2 onto your current operating system.

To install the live upgrade software from the Solaris 9 DVD, follow these steps:

1. Enter the following commands:

```
# cd /cdrom/cdrom0/Solaris_9/Tool/Installers
# ./liveupgrade20
```

   Note: To install from CD #2, enter this command instead:

```
# ./installer
```

2. When prompted to select the type of install, select Custom.
3. On the Locale Selection panel, click the language you want.
4. Choose the software for live upgrade and follow the remaining instructions.

NOTE Live Upgrade can be used to upgrade file systems that are under Solaris Volume Manager or Veritas Volume Manager control, but you cannot use a Veritas volume for the root. For any other file system, you can use a physical disk slice, a Solaris Volume Manager metadevice, or a Veritas volume.

The Solaris Live Upgrade software provides both a command-line interface and a menu interface. The commands are listed and described in Table 5.4.

Table 5.4 Solaris Live Upgrade Commands

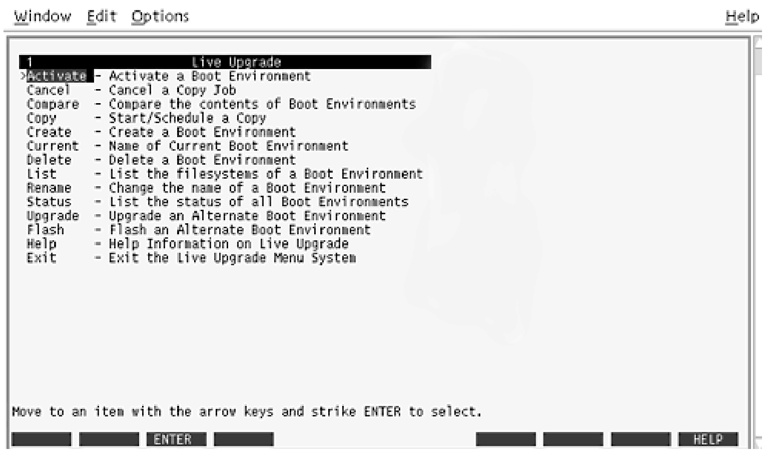
COMMAND	DESCRIPTION
luactivate	Activate an inactive boot environment.
lucancel	Cancel a scheduled copy or a create job.
lucompare	Compare an active boot environment to an inactive boot environment.
lucopy	Recopy file systems to update an inactive boot environment.
lucreate	Create a boot environment.
lucurr	Name the active boot environment.
ludelete	Delete a boot environment.

Table 5.4 *(Continued)*

COMMAND	DESCRIPTION
`ludesc`	Add a description to a boot environment's name.
`lufslist`	List the critical file system for each boot environment.
`lumount`	Enable mounting of all file systems in a boot environment (so that you can modify files while that environment is still inactive).
`lurename`	Rename a boot environment.
`lustatus`	List the status of all boot environments.
`luumount`	Enable unmounting of all file systems in a boot environment.
`luupgrade`	Upgrade an operating environment or install a flash archive on an inactive boot environment.

The menu interface is shown in Figure 5.5.

This has been a hurried introduction to the process of performing live upgrades. We encourage you to read additional material and use this process on a test system before you begin to use it on production systems.



```
1                    Live Upgrade
>Activate - Activate a Boot Environment
 Cancel   - Cancel a Copy Job
 Compare  - Compare the contents of Boot Environments
 Copy     - Start/Schedule a Copy
 Create   - Create a Boot Environment
 Current  - Name of Current Boot Environment
 Delete   - Delete a Boot Environment
 List     - List the filesystems of a Boot Environment
 Rename   - Change the name of a Boot Environment
 Status   - List the status of all Boot Environments
 Upgrade  - Upgrade an Alternate Boot Environment
 Flash    - Flash an Alternate Boot Environment
 Help     - Help Information on Live Upgrade
 Exit     - Exit the Live Upgrade Menu System

Move to an item with the arrow keys and strike ENTER to select.
```

Figure 5.5 Solaris Live Upgrade menu interface.

Patching Your Systems

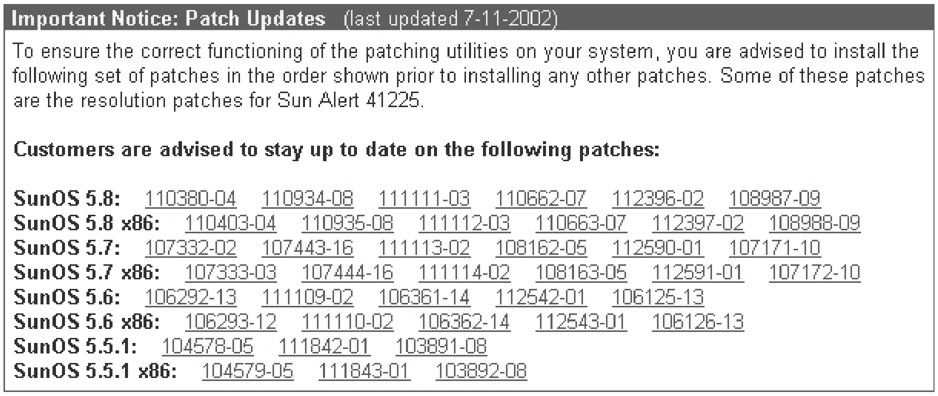
The first thing you want to do once your system installation is complete is to install the accompanying patches. Consider these a part of the operating system; your job is not done until you have installed them.

Patching should be done on both a routine and an emergency basis, depending on the severity of the problems being averted by the particular patches. In this section, we provide some recommendations on how to determine which patches should be applied and sources for information on system vulnerabilities.

Staying Up to Date with Patches

On a routine basis, perhaps once or twice a month, you should check whether new security patches or other recommended patches have been issued, and, if so, install them. This is a sound regimen that is well worth adopting. The best way to keep track of released patches is to subscribe to email alert systems that can send you reports when security holes are discovered and when patches for various problems are released. We suggest that you subscribe to the following alerting services:

- The CERT Advisory Mailing List, available at `http://www.cert.org /contact cert/certmaillist.html.`
- The SANS Network Security Digest, available at `http://www.sans.org`.
- The SunSolve site at `http://sunsolve.sun.com`. It's a good idea to get into the habit of regularly visiting this site. A lot of effort has been put into providing information on system vulnerabilities and patches. Patch reports such as the one shown in Figure 5.6 can guide you to sets of patches that should be installed.
- BigAdmin at `http://www.sun.com/bigadmin/` is also a good place to go for information about Solaris patches. Figure 5.7 illustrates the interface for finding information about patches through BigAdmin.

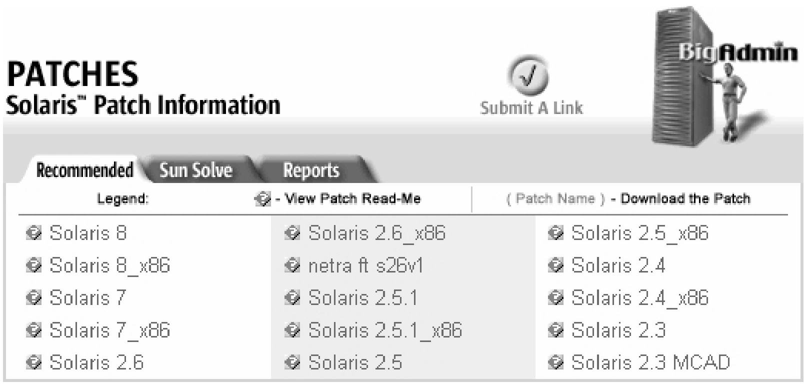

Important Notice: Patch Updates (last updated 7-11-2002)

To ensure the correct functioning of the patching utilities on your system, you are advised to install the following set of patches in the order shown prior to installing any other patches. Some of these patches are the resolution patches for Sun Alert 41225.

Customers are advised to stay up to date on the following patches:

SunOS 5.8: 110380-04 110934-08 111111-03 110662-07 112396-02 108987-09
SunOS 5.8 x86: 110403-04 110935-08 111112-03 110663-07 112397-02 108988-09
SunOS 5.7: 107332-02 107443-16 111113-02 108162-05 112590-01 107171-10
SunOS 5.7 x86: 107333-03 107444-16 111114-02 108163-05 112591-01 107172-10
SunOS 5.6: 106292-13 111109-02 106361-14 112542-01 106125-13
SunOS 5.6 x86: 106293-12 111110-02 106362-14 112543-01 106126-13
SunOS 5.5.1: 104578-05 111842-01 103891-08
SunOS 5.5.1 x86: 104579-05 111843-01 103892-08

Figure 5.6 Patch report from SunSolve.



Solaris 8	Solaris 2.6_x86	Solaris 2.5_x86
Solaris 8_x86	netra ft s26v1	Solaris 2.4
Solaris 7	Solaris 2.5.1	Solaris 2.4_x86
Solaris 7_x86	Solaris 2.5.1_x86	Solaris 2.3
Solaris 2.6	Solaris 2.5	Solaris 2.3 MCAD

Figure 5.7 BigAdmin information on Solaris patches.

TIP Use JumpStart or some other tool to facilitate installation of patches if you are administering a lot of systems.

If you are not sure whether a patch has been installed on a system, you can use the `showrev` command with the -p option; `showrev -p` will list the patches that are currently installed. Pipe the output to `grep` if you are looking for a particular patch. Use this command anytime you want to see all the patches installed on a particular system. Chances are the information won't mean much to you, but it will give you an idea of what's installed.

This is a small example of the output:

```
Patch: 106193-03 Obsoletes: 106350-01 Requires:  Incompatibles:
Packages: SUNWadmap
Patch: 105421-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWapppr
Patch: 105472-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWatfsu
Patch: 105837-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWdtdte
Patch: 105566-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWdtdmn
Patch: 105497-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWoldst
Patch: 105377-03 Obsoletes:  Requires:  Incompatibles: Packages: SUNWbcp
Patch: 105492-01 Obsoletes:  Requires:  Incompatibles: Packages: SUNWcg6
Patch: 105798-02 Obsoletes:  Requires:  Incompatibles: Packages: SUNWcpr
Patch: 105558-01 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWdtdst
Patch: 105338-04 Obsoletes:  Requires:  Incompatibles: Packages:
SUNWdtdst, SUNWdthev, SUNWdtma
```

Why So Many Patches?

Patches are released for three reasons: bug fixes, new hardware support (which sometimes also requires new packages), and new features. As releases stay out in the field longer, more bugs are discovered. Many bugs are found during the development cycle and are fixed in the next release. However, for many customers, an upgrade is out of the question. As a result, more and more patches are released. To protect machines, especially those with public access, patching should be kept up to date.

Patches are fixes to software problems (generally) discovered after distribution and (generally) made available via the Web or an FTP site. Most patches are simply replacement executables; in other words, a new binary file might replace a system process such as the File Transfer Protocol daemon (`ftpd`) when a flaw or a security hole is discovered in the corresponding binary in the previously released operating system.

Occasionally, but rarely, patches are distributed as file insertions used to update existing files rather than replace them. Download and install all pertinent security patches. Recheck the patch list frequently. (There are many places on the Internet to find out what patches are available, such as `http://sunsolve.sun.com`.)

Following are a few things to keep in mind about patching your system(s):

- Not all patches need be installed on every machine. As a matter of fact, it would generally be extremely unwise to install any more patches than you need.
- Even with the extent to which patches are tested, no patch is tested in concert with every other patch. If you install patches irresponsibly, you could end up creating more problems—as a result of incompatible patches—than you were hoping to fix.
- You should always install the patches that come with a new release of Solaris. Consider these patches simply as part of the release that didn't quite make it in time to be incorporated into the files used in the primary installation. These patches *will* have been tested together with the release and will bring you up to the proper and expected operating level.

TIP The patchdiag program that is available on the SunSolve CD and at the SunSolve Web site will automatically compare the patch level of a host against the current Sun-recommended patch set and display any differences. It is a good idea to run this every so often.

Different Types of Patches

A number of descriptors are prepended to the word *patch*: jumbo patch, megapatch, mandatory patch, recommended patch, and security patch all come to mind. The words *jumbo* and *mega* indicate that a number of binaries are being replaced or modified in the same patch file. The term *mandatory* implies that the referenced patch must be installed if you want your system to work properly. *Recommended,* which is the term Sun prefers to use today, means that Sun considers the patches to be universally applicable.

TIP It is not necessary or advisable to install every patch that is released. Read the patch descriptions before installing any patch and make sure that it pertains to your particular environment. Not every patch will be relevant to your systems. Be sure to install the security patches, but, as a general rule of thumb, install only the patches that you know you require.

Obtaining Patches

You can obtain patches from your Sun warranty provider, from the SunSolve Web site, or through BigAdmin (see the URLs for the latter two sites in the "Staying up to Date with Patches" section earlier in this chapter). Patches can be downloaded individually or as part of larger groups called *clusters.* Regular installation of recommended patches is highly recommended, as is use of the tools mentioned in the next section.

Installing Patches

Numerous tools are available today to assist with the installation of patches. The `patchadd` command can be used to install a single patch or a group of patches, as shown in the sample interaction below. Patches that are not relevant to your particular system (e.g., patches for software that isn't installed on your system or patches that have already been installed) will not be installed, so there are no repercussions from being too generous in your patch install specifications. The syntax for the `patchadd` command is:

```
# patchadd -M   directory patch patch ...
```

One way to install a group of patches is to do so in a for loop as shown here:

```
# cd Patch
# for patch in 1*
> do
> patchadd ${patch}
> done
Checking installed patches...
One or more patch packages included in
123456-07 are not installed on this system.

Patchadd is terminating.

 ...

Checking installed patches...
Verifying sufficient filesystem capacity (dry run method)...
Installing patch packages...

Patch number 124578-05 has been successfully installed.
```

```
See /var/sadm/patch/124578-05/log for details

Patch packages installed:
  SUNWlibCx

 ...
```

The authors also recommend the patch tools described in Table 5.5. These tools can facilitate routine application of patches.

Table 5.5 Patch Installation Tools

TOOL	DESCRIPTION
FastPatch by Casper Dik	Fastpatch, (`fastpatch.pl`) is a Perl script that installs Solaris patches as much as five or six times faster than `patchadd`. In addition, the fix-modes script will change the ownership of most files to root. It also has an "undo" switch. For more info on FastPatch, see `ftp://www.wins.uva.nl/pub/solaris/auto-install/`.
PatchDiag Tool	A diagnostic tool from Sun that summarizes your system's patch status. **Note:** You must have a SunSpectrum service account password to download this tool.
PatchDiag to HTML	A Perl script written by Kjetil Homme that converts patch reports produced by PatchDiag tool into HTML.
PatchReport	A Perl script that compares the current patch level of a Solaris system to the recommended patch level and retrieves patches that you need to come up to the proper patch level. PatchReport includes support for Casper Dik's Fastpatch, and can exclude certain patches based on their patch IDs.

Summary

Routine installation of Solaris doesn't require much preparation. Unless you've never done it before, it is a fairly painless process, but one that can take a lot of time. If you're going to install many systems, especially many systems of the same basic type, using one of the newer installation tools can save you a lot of time and trouble.

The use of *Web Start Flash* archives present the largest return on your preparation time investment if you are installing many systems with the same configuration. By creating one or more flash archives from master systems, you can create numerous clones, and in much less time, than any other type of installation. By creating archives that contain portions of master systems, you can layer your clones for additional flexibility. Once built, Web Start Flash archives can be used with any of the installation methods available today.

Live upgrades allow you to perform the bulk of a system upgrade while the system is still being used. By setting up second copies of critical file systems that are then upgraded and activated only when you're ready to cut over, you reduce the downtime of a system to little more than the time it takes the system to reboot.

Web Start installations provide an interface that allows you to move backward and forward through the installation process and consult system documentation during the installation process.

The *suninstall* tool provides a command-line interface, but can be used with a preconfigured `sysidcfg` file to speed up the process by reducing the amount of information that the system must collect from the installer at installation time.

JumpStart (see Chapter 6) continues to provide a rule-based installation method that offers the advantages of regularity and uniformity if you are going to install more than a handful of systems, especially if they'll be serving the same role and are similar systems. Use of JumpStart requires that you prepare a JumpStart server and profiles for the systems to be installed. Even so, on a large network, you will probably be very glad that you made the initial investment.

Patching is a necessary part of system administration. Here are some tips:

- Periodic application of appropriate patches will keep your systems more secure and is so important that it should become a routine.
- We suggest patching your systems as needed once or twice a month. Don't install patches unless you know that you need them, however. Doing so can create new problems, because patches are not tested in conjunction with all other patches—except for those that arrive with new releases of Solaris.
- Using JumpStart to install patches will simplify the process of installing patches on numerous systems.

CHAPTER

6

Exploiting JumpStart and Web Start Flash

The single most energy-saving strategy that any system administrator can adopt is to enforce a uniformity in the configuration of his or her systems. By doing so, most of the work involved in managing these systems and the networks on which they reside is greatly simplified. If you know how a service is configured, for example, you know how it's configured on *all* systems. By automating the installation of systems, you can greatly improve the uniformity of the systems you manage. You reduce the information that must be supplied at the time of installation to little or nothing and remove the monotony of the task—both good steps toward creating reliable and controllable systems. As a result, you can expect to spend more time solving real problems and less time unraveling how things were supposed to have been set up.

JumpStart is a tool that was derived from Sun's internal need to install its inventory of systems. When used properly, it helps to facilitate uniformity by automating the installation process. It reduces the time and effort involved in installing systems in proportion to the number of systems that need to be installed and can also be used to automate installation of patch releases and application software. JumpStart requires an up-front investment of time, planning, and training, but once set up JumpStart pays off each time you use it.

With the addition of Web Start Flash (which first appeared in a release of Solaris 8), JumpStart takes on another level of efficiency for repetitive installations. It is now possible not only to create profiles that automate installation, but also to create archives that contain all the files needed to set up and configure a system to play a particular role (such as a database server or a Web server). By using JumpStart and Web Start Flash together, you do most of the work of configuring systems when you plan the services that they will provide. This includes the installation and configuration of application software as well as the operating environment. The installations, which occur in a second phase, are by then fairly rote. Any site that must repeatedly install hundreds or thousands of systems will clearly find itself with a more streamlined and reliable process if it uses these tools.

This chapter covers JumpStart and provides information on how to incorporate the use of Web Start Flash archives into your JumpStart installation process.

JumpStart and Web Start Flash Terminology

Before we get into the describing how JumpStart servers are set up and Web Start Flash archives created, let's quickly review the terms used in dealing with this technology.

Rule. A line of text that identifies the systems to which it will be applied, along with the name of the profile that will be used and the (optional) begin and finish scripts.

Rules file. A text file that contains a rule for each group of systems you will install using JumpStart.

Profile. A text file that describes how each system in a particular group is going to be installed (e.g., what installation methodology is to be used and what software cluster is to be installed).

Profile diskette. A diskette that contains the rules file and profile, which are required in the rare situations in which you might want to JumpStart on a standalone system.

Profile server. A server that provides rules and profiles for JumpStart installations.

Web Start Flash archive. A snapshot of a system that has been set up and configured for cloning.

These terms will be described in more detail throughout this chapter.

Custom JumpStart

Custom JumpStart provides for the automated installation of systems according to directions stored in simply formatted text files called *profiles.* What Custom JumpStart does for you is this: It allows you to describe the setup of systems—specifying file system layout, software packages, and other installation parameters—and then use these descriptions to install as many systems as fit the particular profile in a streamlined fashion.

Preparing for an individual client or a group of clients to be installed with the same basic layout requires you to do the following:

1. Build a profile that describes the layout and configuration of these systems and makes it accessible to the clients being installed.
2. Identify the systems for which this profile applies (i.e., create a rule).
3. Provide access to the installation media.
4. Provide boot support for each of the specific clients to be installed.

We explain each of these roles in detail in the "Server Roles" section later in this chapter.

Web Start Flash

Web Start Flash should be thought of as an extension to JumpStart, not a competing technology. Though Web Start Flash archives can be used with any of the installation methods, they provide the most advantage when combined with JumpStart. You can essentially automate the installation and configuration of systems, including not just the operating system but application software as well. As appealing as this may seem, however, there is no particular advantage to using Web Start Flash in a site where every system is different. Only when you can set up a single system and clone it repeatedly does automating the process become worth the time it takes you to prepare for this installation technique.

Where a traditional JumpStart installation uses a JumpStart profile to specify the software packages or software clusters to be installed and the `pkgadd` command to sequentially install each package and update the package database, JumpStart with Web Start Flash bypasses this process and installs files directly from the archive.

As explained in Chapter 5, Web Start Flash involves preparing a master system, capturing that system's files and configuration, cloning other systems to be virtually identical to this master system, and then customizing the clones to give them each a network identity of their own. As you might expect, there is an ideal time at which to prepare a Flash archive. That time is immediately after a system has been installed and configured to your specifications, but before it has been put into use. You will not, for example, want to duplicate home directories, log files, or any other information that relates to the master system's use as opposed to the role that it plays on your network.

The best approach to using Web Start Flash is to identify master systems to serve as models for such roles as database server, application server, Web server, file server, test system, and end-user workstations and then clone these systems as soon as they are properly set up. We also recommend storing all Flash archives on your JumpStart server after creating them on newly installed systems. This is a logical place for them to be stored because your JumpStart server plays such an important role in installing systems.

Web Flash archives can be used with any of the installation methods as long as you have the information about the particular archive available at the point in the installation where the software prompts you to provide it.

Installing from Web Flash archives is much faster than installing from any other form of installation media. When software is installed from an archive, individual packages are not installed in sequential fashion. Instead, the contents of the archive are installed in one step. This avoids the time-consuming step of updating the package map multiple times.

Installing from a Web Flash archive is much more like restoring from backup than it is like installing a system from installation media. The process involves reading all files included in the archive and installing them in place. Files related to the installation of packages, for example, are simply dropped into place, as are all other files, rather than being updated package by package during a formal installation process. As you can imagine, this speeds up the installation considerably.

Because a Flash installation overlays all files from a previous installation, it can only be used for a full installation.

NOTE Upgrades, in which only those files modified by the new software release, are not possible with Web Start Flash.

In spite of the fact that Flash archives are used to create clones of the master system, some changes can be incorporated into a Flash installation. For example, if your target system has more disks than the master system, you can configure your installation profile to partition the disks so that a file system (such as `/opt` or `/var`) that is not separate on the master system is installed on a separate disk on the target system. After the disks on the target system are partitioned, the files loaded into these file systems will be placed in the appropriate partitions just as if you were extracting them from a tar file.

Further, customization of clone systems can be done by hand or by running a script that you prepare for this purpose. If you prepare a script to make the changes needed on the individual clones, you can elect to run this script by hand or as a *finish script* within your JumpStart installation. One of the benefits of automating installation to the fullest extent possible is that you can separate installation planning from the installation process. You can have, for example, one small team that lays out the configuration of a network and ensures that all settings are proper and compatible, and another that installs the systems and ensures that each system works as intended. This separation of focus works well in many organizations and is a natural "checks-and-balances" technique.

The Format of Archives

Archives are built in a special format that contains various sections. An archive must have at least the three mandatory sections, but may also contain optional user-defined sections. Each of the sections is described in Table 6.1.

Table 6.1 Web Start Flash Archive Sections

SECTION	KEYWORD	USE
Archive Cookie Section	cookie	Contains a cookie (i.e., a magic number) that identifies the file to the system as a Web Start Flash archive.
Archive Identification Section	identification	Keyword/value pairs that describe the archive.
User-Defined Sections	*anything*	Optional line-oriented information that is included at the request of the user. All lines end with a newline character (octal 12). Binary data must be encoded with base 64 or other algorithm that changes the values into ASCII.
Archive File Section	archive	The files stored from the master system.

Each section begins with the section_begin keyword and ends with the section_end keyword. Keywords that can be used in the identification section are described in Table 6.2.

Table 6.2 Web Start Flash Identification Section Keywords

KEYWORD	MEANING
content_name	Identifies the archive to the Web Start Flash archive deployment utilities. The value can be up to 256 characters. The name should be meaningful and convey the purpose of the archive.
creation_date	The date the file was created in the format YYYYMMDDhhmmss. The default is set to GMT.
creation_master	The name of the system on which the archive was created. The flarcreate command will issue the uname -n command if this information is not specified by the user.
content_type	A user-supplied category for the archive. The name should be meaningful.
content_description	A description of the contents of the archive. There is no length limit to this field.
content_author	A description of the individual who created the archive. Sun suggests that name and email address be included.
content_architectures	A comma-delimited list of the system architectures that the archive supports. This information is generated when the archive is created. When the archive is used, this value is checked against the architecture of the system being installed.
creation_node	The value returned by uname -n on the master system or the contents of the nodename file if the root directory for the archive is not /.
creation_hardware_class	The value returned by uname -m on the master system or the string UNKNOWN if the root directory for the archive is not /.
creation_platform	The value returned by uname -i on the master system or the string UNKNOWN if the root directory for the archive is not /.

(continues)

Table 6.2 Web Start Flash Identification Section Keywords *(Continued)*

KEYWORD	MEANING
`creation_processor`	The value returned by `uname -p` on the master system or the string UNKNOWN if the root directory for the archive is not /.
`creation_release`	The value returned by `uname -r` on the master system or the string UNKNOWN if the root directory for the archive is not /.
`creation_os_name`	The value returned by `uname -s` on the master system or the string UNKNOWN if the root directory for the archive is not /.
`creation_os_version`	The value returned by `uname -v` on the master system or the string UNKNOWN if the root directory for the archive is not /.

Rebuilding from Scratch

In the Sun BluePrint entitled *JumpStart Technology: Effective Use in the Solaris Operating Environment*, authors Howard and Noordergraaf suggest storing Flash archives off-site in addition to storing them locally. Because these archives would allow you to recreate systems used in your business, they would make it possible for you to quickly provide for continuity in the case of disaster. For example, using these Flash archives you could set up your database and Web servers at another site in a matter of hours. This kind of planning should be considered by all companies serious about guaranteeing their site's availability.

Creating and Storing Archives

To create Web Flash archives and make them available for subsequent JumpStart installations, you would need to do the following:

- Install and configure the master system
- Create the archive with the `flarcreate` command
- Move the archive to the JumpStart server (or server of your choice)

Each archive should be given a name that indicates its contents. In addition, your naming scheme should be consistent and easy to understand. For example, you might name your Flash archives to indicate the OS release and the primary function of the target systems. For example:

- Solaris 9 Web server
- Solaris 9 database server
- Solaris 8 test box

These names will make it easier for installers to select the proper archive for the job and to anticipate what file names to look for. When an archive is ready for use in configuring test boxes for Solaris 9, for example, installers will know to look for “Solaris 9 test box” on the JumpStart server.

To create a Solaris 9 test box archive on a master system set up as a test box (i.e., a system dedicated to quality control and load testing of software under development), you might use a command such as this:

```
# cd /var/tmp
# flarcreate -n "Solaris 9 test box" \
> -a "jdoe@foo.com" \
> -R / \
> -x /var/tmp \
> /var/tmp/Sol9_testbox.archive
Determining which filesystems will be included in the archive...
Determining the size of the archive...
The archive will be approximately 650.33MB.
Creating the archive...
Archive creation complete.
```

This command carefully omits the archive itself and any other files stored in `/var/tmp`. The -x argument excludes the `/var/tmp` directory while -R causes the remainder of the system’s directories to be recursively traversed when files are collected for inclusion. This command also provides both an archive name and a file name that describe the contents of the archive. Options for the `flarcreate` command are described in Table 6.3.

Table 6.3 Options for the flarcreate Command

OPTION	DESCRIPTION
Required	
-n name	The name (content_name) of the archive (this “option” is actually required).
Compression	
-c	Compress the archive with the compress command.
Directories and Sizing Info	
-R root	Create the archive, starting at the “root” specified; otherwise, it starts from the actual system root (/).
-S	Do not include sizing information within the archive.
-H	Do not generate hash identifier.

(continues)

Table 6.3 Options for the flarcreate Command *(Continued)*

OPTION	DESCRIPTION
Excluding Directories	
-x exclude	Exclude the specified file from the archive, relative to the specified root if -R is used as well.
User-Defined Sections	
-u section	Include the user-defined section specified (can be a space-delimited list).
-d dir	Specify the directory to be used with the -u.
Tape Archives	
-t	Create the archives on the specified tape device.
-p position	Specify the tape position when the above option is used.
-b blocksize	Specify the block size to use (otherwise uses the default).
Files	
-f file_of_filenames	Add the files listed in the specified file to the archive.
-F	Use *only* the files specified in the -f command in the archive.
Archive Identification	
-U key=value	Include keywords specified in the identification section of the archive.
-i date	Use the date specified as the creation_date in the archive (otherwise it uses the current date).
-m master	Use the name specified as the creation_master in the archive (otherwise it uses the output from the *uname -n* command).
-e description	Use the description provided as the content_description in the archive.
-E description_file	Set the content_description from the file specified (incompatible with the above option).
-a author	Use the name specified as the content_author in the archive.
-T type	Use the type indicated as the value of content_type in the archive.

Following creation of an archive, it is good practice to use the `flar -i` command to ensure that the contents of the archive appear to be correct. The archive created in the preceding example would provide information that looks roughly like this:

```
# flar -i /var/tmp/Sol9_testbox.archive
archive_id=06b03a0c0ffee09015090638e0836a92
files_archived_method=cpio
creation_date=20020923112233
creation_master=dragonfly
content_name=Solaris 9 test box
files_compressed_method=none
files_archived_size=650223058
content_author=jdoe@foo.com
content_architecture=sun4u
```

When this archive is used in a JumpStart profile, it might look like this:

```
install_type    flash_install
archive_location nfs://10.4.3.2/jumpstart/archives/Sol9_testbox.archive
partitioning    existing
```

Notice that the installation method in the above profile is specified as "flash_install."

Splitting an Archive

The `flar` command, when used with a -s option, splits an archive into its sections, each of which is then stored in a separate file. Each section is stored in the current or specified directory. Each file that is created is named after the section of the archive that it contains.

The syntax for the `flar -s` command is:

```
flar -s:split [-d directory] [-u section] [-f archive] [-S section] [-t
[-p position] [-b blocksize] ] filename
```

The `-d directory` option allows you to specify an alternate location (other than the current directory) for the archive. The `-u section` option is used to specify which sections are to be copied. If it is not used, all sections are copied. If it is used, the cookie, identification, and archive sections are copied, along with any other sections specified. If the `-f archive` option is used, the archive is extracted into a directory by the specified name rather than into a file by the specified name. If the `-S section` option is used, only the specified section is copied.

Combining Archives

The `flar` command used with the -c option enables you to create an archive from sections that have been separated. The command expects each section to be in a file named after the particular section. The standard sections must be available for the command to work; other sections are optional.

The syntax of the `flar -c` command is:

```
flar -c:combine [-d directory] [-u section] [-t [-p position] [-b
blocksize] ] filename
```

The `-d directory` option allows you to specify an alternate location (other than the current directory) for the archive. The `-u section` option allows you to select sections to be included in the archive. If this option isn't used, all sections will be included. Otherwise, the standard sections—cookie, identification, and archive—will be included along with the sections specified.

Layered Archives

Layered archives provide more flexibility in configuring systems. In other words, instead of installing an entire system from a single archive, you can install a system from a series of archives. For example, the basic operating system might be installed from one archive and a series of applications from another. You might create archives for specific types of systems—such as database servers and Web servers—that are applied after installing Solaris.

Server Roles

It is important to note that although each of the server roles we are about to describe is fairly distinct, a single server can be used to provide any or all of these services. In fact, many system administrators insist on maintaining a single server that fills the roles of installation, boot, and profile server.

NOTE If the systems that you manage are on different subnets, you will need to have a boot server (if not an installation server) on each subnet.

Further, any of these roles can be served by systems that are already up and running on your network. Although we advocate a separation of major network services to the extent that this is reasonable (so that problems with one critical network service do not affect the operation of others), some doubling up of services makes a lot of sense. Your NIS/NIS+ server might, for example, be the ideal candidate to also serve as your profile server. An existing file server might conveniently serve as a boot server. Combining related services on a single server may, in fact, turn out to save you quite a bit of time and effort. Just understand that these services can have an impact on each other when they share CPU and disk space.

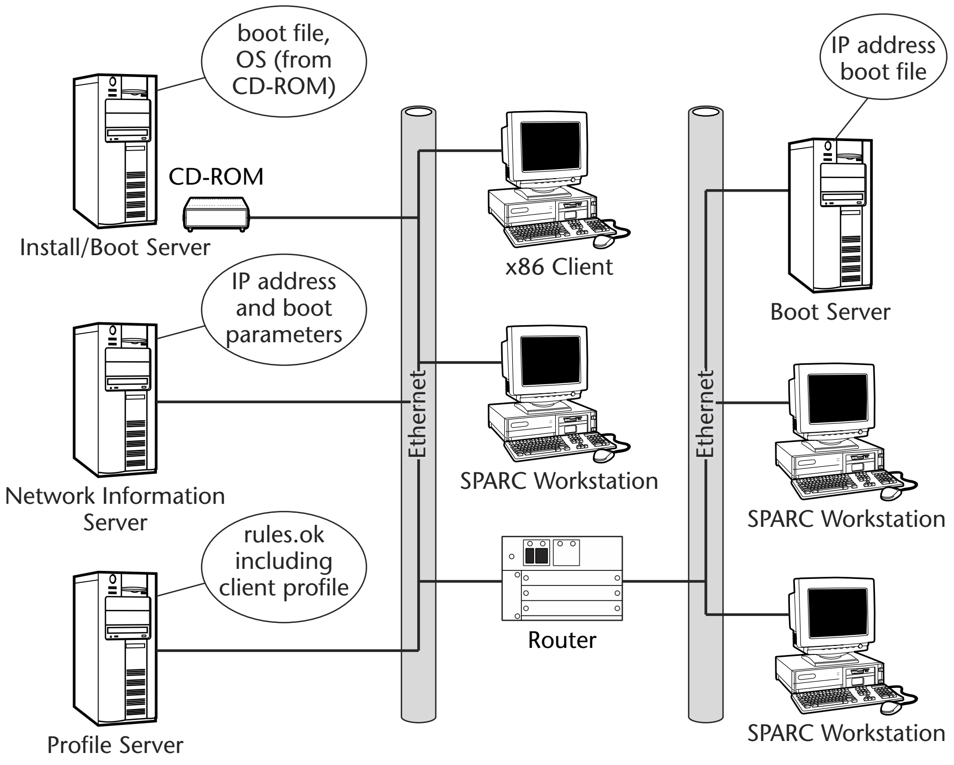


Figure 6.1 Server roles.

Figure 6.1 depicts the different roles that servers play during installation on a network with subnets. Note the position of the two boot servers relative to the clients, and the doubling up of server roles on one system.

Installation Servers

An installation server provides the software and media for the installation. In the past, this was generally made available through a Solaris installation CD, which was mounted and exported so that the client being installed or upgraded could access it. Increasingly, an installation server will have the files from the Solaris installation CD copied onto its hard disks or will provide these files through a Web Start Flash archive to improve performance and decrease the reliance on installation media. An installation server can be located anywhere on your network; however, it is generally better (because of the possible impact on network traffic) to have the installation server in close proximity (with respect to network cabling) to the client being installed. Figure 6.2 depicts an installation server.

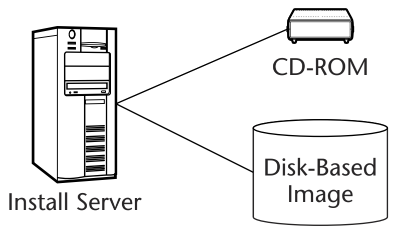


Figure 6.2 Installation server.

NOTE **For most organizations, the installation server and JumpStart server are the same system.**

Boot Servers

A boot server is a system that supports the client's initial booting so that the installation process can begin. In other words, it provides the boot software to the client. The role of a boot server during a Solaris installation is much like the role a boot server plays in booting a diskless client. The client, booted with a command such as `boot net` that diverts its attention away from its hard disk, needs to first establish its identity with respect to the network, and then obtain the boot software that will allow it to complete the installation process.

Unless the clients you are installing are on a different subnet than your install server, you do not need to set up a separate boot server. The installation server already contains the files that the client needs to boot. For clients on subnets that do not have local installation servers, however, a boot server must be provided. Clients that boot from a local boot server can still obtain the installation software from a remote installation server.

Profile Servers

A profile server is a server that houses the profiles for each of the various system configurations you want to install using JumpStart. Because the amount of information that it provides during installation is minimal, although critical, a profile server can be located anywhere on your network.

To create a profile server, you create a directory, share it, and populate it with the client profiles and the rules file that determines when each profile is to be used. Figure 6.3 depicts a profile server.

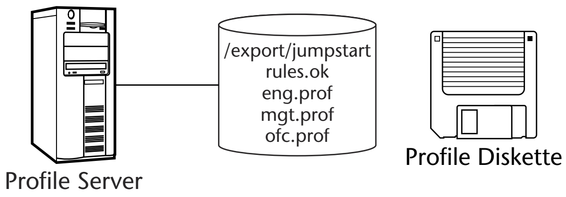


Figure 6.3 Profile server.

An alternative to a profile server, a profile diskette, can be created in essentially the same manner. In short, you format a diskette, create a file system, and add the profiles and rules files. Profile diskettes for use with SPARC systems must contain a UFS or pcfs file system. Profile diskettes for x86 systems must be copies of the Solaris diskette that also contain the profiles and the `rules.ok` file.

Network Information Server

A network information server (or NIS), for the purposes of a Custom JumpStart installation, provides a mapping not only of hostnames to IP addresses but also the mapping of hardware addresses to IP addresses needed for booting. It also supplies boot parameters for clients booting over the network. We're using the term *network information server* somewhat loosely here. This information can be derived from system files (such as the familiar `/etc/hosts` and the `/etc/bootparams` files) rather than supplied by a formal name/information service such as NIS or NIS+. Figure 6.4 depicts a NIS server.

Configuring Your Site for JumpStart

Setting up your site in order to maximize the utility of JumpStart involves preconfiguring a number of services so that the per client investment at installation time is minimized. In order to make use of the JumpStart technology, you will need to have servers ready to assume the server roles that we just described. Figure 6.5 shows an approach to configuring JumpStart that starts with those services that we are assuming are already in place (and are required by JumpStart) and ends with those steps that you must take to invoke a JumpStart installation on the client.

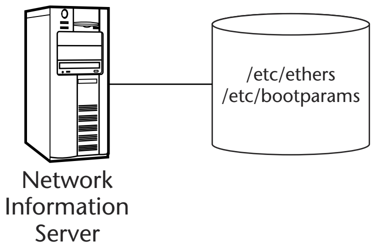


Figure 6.4 NIS server.

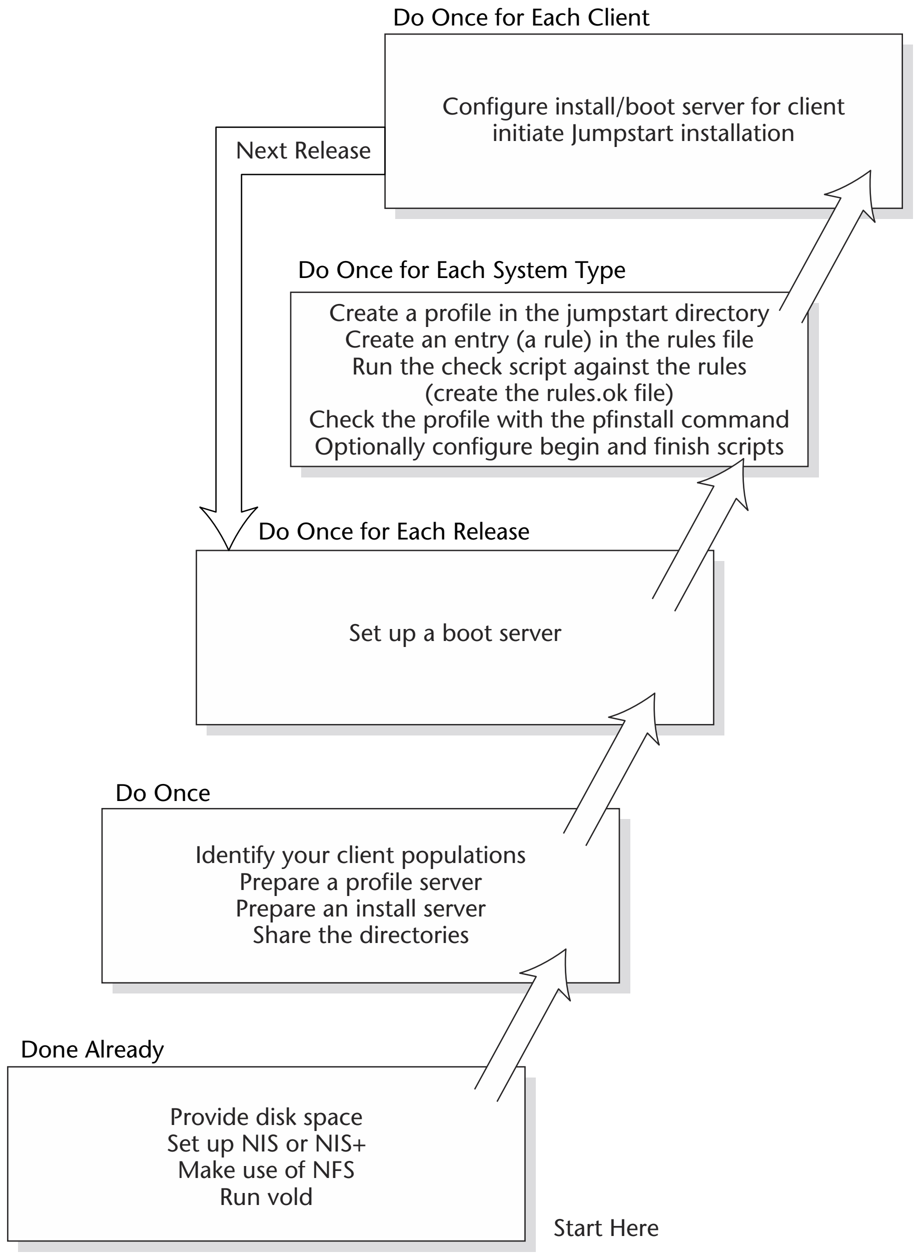

Do Once for Each Client
Configure install/boot server for client
initiate Jumpstart installation
Next Release
Do Once for Each System Type
Create a profile in the jumpstart directory
Create an entry (a rule) in the rules file
Run the check script against the rules
(create the rules.ok file)
Check the profile with the pfinstall command
Optionally configure begin and finish scripts
Do Once for Each Release
Set up a boot server
Do Once
Identify your client populations
Prepare a profile server
Prepare an install server
Share the directories
Done Already
Provide disk space
Set up NIS or NIS+
Make use of NFS
Run vold
Start Here

Figure 6.5 JumpStart steps.

Tasks That Have Already Been Performed

A certain portion of the preparation for JumpScript installation is, undoubtedly, already part of your network configuration. Most likely, for example, you already have

NIS or NIS+ support and can provide boot parameters, time zones, regions, and maps in the information service or system files. We are assuming that you meet the following requirements:

- Have disk space available for profiles, client support (boot files and client root and swap), and so on
- Are prepared to provide name services to newly installed systems
- Are well acquainted with NFS sharing and mounting
- Are running volume manager (vold)

Tasks That Need to Be Performed Once

The most intense effort of Custom JumpStart installation involves creating the profiles and providing them, along with the installation media, disk images, or Flash archives, to the clients during the boot process. Most sites will set up servers to fill the roles of boot server, install server, and profile server on a more or less permanent basis, updating these systems as needed. To prepare for later steps in your JumpStart configuration, you must do the following:

- Prepare an inventory of your systems
- Identify and preconfigure a profile server
- Identify and preconfigure an install server
- Share the required directories

Preparing a Client Inventory

A good first step in preparing for Custom JumpStart is to prepare a descriptive inventory of the systems you will be installing. Remember that to reap the greatest benefit, you need to organize your systems into groups that can share profiles. The effort that you put into compiling an accurate system inventory will pay off later in the ease with which these systems can be managed and upgraded. Your inventory should include the type of system, model, disk size, memory, and a description of the primary user or primary role the system plays. Development systems will require a different suite of software and tools than systems that only run packaged applications.

Preparing a Profile Server

Selecting a system to serve as profile server does not involve a lot of effort. Almost any system will serve the purpose quite well. The disk space and operational demands placed on a profile server are minimal. After you've selected a system to play this role, you then create the directory that will house the profiles and other JumpStart files and share it. Later, we will populate this directory with profiles and other JumpStart files corresponding to the distinct client populations isolated in the system inventory.

```
# mkdir -m 755 /jumpstart
# echo "share -F nfs -o ro,anon=0 /jumpstart" > /etc/dfs/dfstab
# shareall
```

Identify and Preconfigure an Installation Server

To set up an installation server, you must both set up access to the Solaris CD images that you will use for installing systems and configure the installation server to boot the install clients (i.e., any system that is to be installed from this server). When you first set up an installation server, you should set aside a large enough partition to accommodate the number of releases that you intend to support at any time. We suggest that you reserve between 3 and 4 GB for each release.

With each release of Solaris, you will install a new CD image on the server (see the following section).

Share the Required Directories

You must also share the directories on the server that contain the CD images so that clients can mount them during an installation. Add the directories to your `/etc/dfs/dfstab` file. Your entry might look like this:

```
share -F nfs -o ro,anon=0 /jumpstart
```

You can share the directory interactively by issuing the `shareall` command.

Tasks That Need to Be Performed for Each Release

Every time you receive a new release of Solaris, you'll need to update a number of your servers to prepare for installation upgrades. You will need to do the following:

- Provide the new installation software on your installation server
- Update any boot servers with the new boot files
- Copy the example JumpStart files from the Solaris installation media with a command such as this:

  ```
  # cp -r <media>/Solaris_9/Misc/jumpstart_sample/* /export/jumpstart
  ```

Each release of Solaris requires a larger amount of disk space than the prior version. We suggest that you be generous in providing enough disk space that your JumpStart server will be able to provide service through any number of releases. We recommend that you set aside between 3 and 4 GB of space for each release.

Update Your Installation Server

You can update your installation server by extracting installation files from the Solaris CDs or by copying Flash archives created on master systems to a directory created for this purpose.

Installing from CDs

To update your installation server, you can simply choose to mount the new installation CD or copy the installation software onto your hard disk from the CD or from another server. In general, copying the installation image onto your hard disk will provide for much faster installation; reading from disk is considerably faster than reading from a CD-ROM. Keep in mind, however, that a disk-based copy of the installation software for Solaris 9 will require as much as 3 GB of disk space and will take a long time to copy onto your hard disk. In fact, this is one of those tasks that is best started before you leave for the night and checked the following morning.

If the volume manager (vold) is running, simply loading the media is all that's required to mount it. Whether you're using a CD or a disk-based installation image, the file system on which the file resides must be exported.

Use the `setup_install_server` command if you want to create a disk-based image of the installation media. Disk-based installation provides the added benefit of allowing you to effect a prompt-free installation through modification of only a few files (which, clearly, cannot be done using a CD). Once the installation files are located on your disk, site-specific information can be inserted into the appropriate files.

With the Solaris CD #2 in the drive, create a directory to hold the CD image and copy the files using the commands shown below. You will find the `setup_install_server` command on the installation CD in a Tools directory on slice 0, as shown here:

```
# mkdir -p /export/install
# cd /cdrom/cdrom0/s0/Solaris_9/Tools
# ls set*
setup_install_server
```

To run the command, you need to provide the directory to be used for the disk-based installation software:

```
# ./setup_install_server /export/install
```

After you run this command, you will have an image of the CD on your server that will contain files required for client installation. A sample directory listing is shown here:

```
installserver% pwd
/export/install/Solaris_9/Tools/Boot
installserver% ls -l
total 40
drwxr-xr-x   2 root      sys          512 Oct  6 13:38 a
lrwxrwxrwx   1 root      other          9 Feb  1 17:23 bin -> ./usr/bin
drwxr-xr-x   2 root      sys          512 Oct  6 13:38 cdrom
drwxrwxr-x  10 root      sys         1024 Oct  6 13:43 dev
... and so on
```

Share the installation server's files by adding a line such as this to the `/etc/vfstab` file:

```
share -F nfs -o ro,anon=0 -d "install server" /export/install
```

Make sure NFS is running or start it with this command:

```
# /etc/init.d/nfs.server start
```

And then share the file system with the `shareall` command:

```
# shareall
```

Because Solaris 9 is distributed on more than one CD, you will need to use the command shown below to add software from CD #2:

```
# ./add_to_install_server /export/install
```

If users are going to be using the Solaris Web install without CDs in their local drives to install their systems, you will also need to load the Web interface software to support this. You will use the `modify_install_server` command as shown below:

```
# cd /cdrom/cdrom0/s0
# modify_install_server -p /export/install <installer_miniroot_path>
```

You can maintain more than one release of the installation software if you want to be able to install clients at different release levels. As noted in the preceding example, the `setup_install_server` command creates a directory specific to the release being installed. A different release of the software created with the same command (and a different CD in the drive) would create a sister directory in `/export/install`.

Tasks That Need to Be Performed Once for Each System Type

For every distinct population of clients, you will need to prepare a profile describing the way these systems should be installed and make it available to the clients. The steps involve the following:

1. Creating a profile for each group of similar systems (see the next section).
2. Testing the profile using the `pfinstall` command to determine if it will work as you intend (e.g., if the system has enough disk space).
3. Adding each profile to the `rules` file (see the next section).
4. Validating the `rules` file with the `check` command. If the check is successful, a `rules.ok` file will be generated.
5. Exporting the `/export/jumpstart` directory with the following command:

   ```
   share -F nfs -o ro,anon=0 /export/jumpstart
   ```

Creating a Profile

Creating a profile involves describing a group of the systems that you identified when you prepared your system inventory and outlining how these systems should be configured and installed.

Give your profiles names that clearly describe the type of system or its use (e.g., `test-box.prof` or `web-server.prof`). Store each profile as a separate file, and save it in the jumpstart directory on your server (or on a diskette if you are going to use a profile diskette).

Many sites will start the process of building profiles by copying the sample profiles from the installation CD. This is a good idea as long as you take the time to understand each line in the file. The following listing shows a sample profile, and the keywords available for JumpStart profiles in Solaris 7 are shown in Table 6.1.

```
# test-box-client
# keywords            values
install_type          initial_install
system_type           standalone
cluster               SUNWCprog
filesys               c0t0d0s0 auto /
filesys               c0t3d0s1 512 swap
filesys               any auto usr
dontuse               c0t2d0
package               SUNWolman delete
package               SUNWxwman delete
package               SUNWoldem add
package               SUNWxwdim add
```

The file should contain one keyword per line and the first keyword to appear should be the `install_type` keyword.

The first two lines in the preceding code are comments that help identify the use of this profile. The next two lines describe the type of installation and the type of the system. The three `filesys` lines specify that the system should use the 0th partition on disk 0 for the root file system, that the system should create a 512 MB swap partition on the system disk, and that the system should build the `/usr` file system on any disk and allocate file space automatically. The `SUNWCprog` line is requesting the developer cluster. The `dontuse` keyword specifies that the disk listed should not be affected in any way by the installation.

Profiles should be root-owned and their permissions set to 644.

The Profile Syntax

The syntax for creating profiles uses very simple keyword/value pairs. The number of options available, on the other hand, is substantial. Table 6.4 provides a description of each keyword along with examples of possible values.

Table 6.4 Profile Keywords

KEYWORD	DESCRIPTION

`archive_location`	Specifies the retrieval method and the location of a Web Start archive. It should be specified with the syntax: `archive_location retrieval_type location_specifier`, where the retrieval type can be any of the following: `nfs`, `http`, `local_tape`, `local_device`, or `local_file`.
	
	Examples include:
	`archive_location nfs archserver:/opt/ archives/ Sol9_testbox.archive`
	
	`archive_location http archserver:80 /archives/ Sol9_testbox.archive`

`backup_media`	Defines media that will be used if, due to space problems, files need to be backed up (moved aside) during the installation—this keeps you from running out of room.
	
	`local_tape` `/dev/rmt/#`
	
	`local_diskette` `dev/rdiskette/#`
	
	`local_filesystem` `/dev/dsk/c#t#d#s# or /export/spare`
	
	`remote_filesystem` `remhost:/export/spare`
	
	`remote_system` `user@host:/directory (uses rsh)`

`boot_device`	Specifies where the system's root file system will be. This will also be the boot device. With a third argument, specifies what to do with the current EEPROM boot disk value. `c#t#d#s#` or (x86) `c#t#d#` or `c#d#` `existing`—Current boot device `any`—Installation program chooses, most likely existing boot device The EEPROM argument can be set to update (modify EEPROM entry for the boot device) or preserve (leave EEPROM as is).

Table 6.4 *(Continued)*

KEYWORD	DESCRIPTION
`client_arch`	Specifies the client hardware—SPARC or x86. Must be used if the system is a server that will support a client architecture different from its own. Can be any of the following: `sun4d`, `sun4c`, `sun4m`, `sun4u`, or `i86pc`. This option can be used *only* if the `system_type` is set to server.
`client_root`	Defines the amount of space to allocate for client root file systems. This option can be used *only* if the `system_type` is set to server. If this value is not specified, the installation software will allocate 15 MB for each client (determined by the `num_clients` value). The space allocation is used to build the `/export/root` file system.
`client_swap`	Defines the amount of space to allocate for client swap space. This option can be used *only* if the `system_type` is set to server. If this value is not specified, the installation software will allocate 32 MB for each client (determined by the `num_clients` value). The space allocation is used to build the `/export/swap` file system.
`cluster` (adding software groups)	With a single argument, this keyword specifies the software group to install on the system. The options include `SUNWCreq` (core, the minimal installation cluster), `SUNWCuser` (end user support), `SUNWCprog` (the developer installation), `SUNWCall` (the entire distribution), and `SUNWCXall` (the entire distribution plus OEM support, available for SPARC systems only).
`cluster` (adding/deleting clusters)	With both a cluster name and an add or delete argument, this keyword specifies a particular software cluster that should be added to the software group being installed or omitted from it. This keyword/value set is required only if you want to tailor your installation more finely than the software groups. If neither `add` nor `delete` is specified, `add` is assumed.

(continues)

Table 6.4 Profile Keywords *(Continued)*

KEYWORD	DESCRIPTION
`dontuse`	Specifies a disk that will not be used by the installation software. If this command is not used, all disks on a system are assumed to be available for use. This command, therefore, should be used to preserve a disk that you do not want the installation to affect. Applications and data files or Web sites and their log files are some examples of files that you might want to preserve with this command—if they are on a disk that you do not need for the new release. You cannot use both this keyword and the `usedisk` keyword in the same profile.
`fdisk` (not applicable for Solaris 9 without x86 support)	Specifies how the installation will affect the fdisk partitions on an x86 system. If the intention is to have a dual-boot client, you want to be sure that existing partitions are not overwritten during the install. If this keyword is not used at all, existing partitions are overwritten. Three arguments are available with this keyword: `diskname`, `type`, and `size`. The disk argument lets you choose which disk to use. The most common format specifies the disk by controller, target, and disk (e.g., `c0t3d0`) or controller and disk (e.g., `c0d0`). Alternately, you can specify `rootdisk` (i.e., the installation software will determine the disk to use) or all (i.e., all of the selected disks). The type argument determines the type of fdisk partition to be created or deleted. Three values are available: `solaris` (a Solaris fdisk partition), `dosprimary` (a primary DOS fdisk partition), and `DDD` (an integer fdisk partition). To overwrite the existing fdisk partitions explicitly, use `fdisk all solaris maxfree`. One choice for preserving the existing fdisk partitions is `fdisk rootdisk solaris maxfree`.
`filesys`	There are two uses for the filesys keyword: to specify which file systems are mounted automatically when the system boots and to create local file systems. Both uses affect the `/etc/vfstab` file on the client. If the `filesys` keyword is used with a remote filesystem, the first type is assumed. The remaining arguments (following the `server:/filesystem` argument) are server IP address, local mount points, and file system options (e.g., `-o ro`). If the `filesys` keyword specifies a local filesystem, the first argument is the slice (e.g., `c0t0d0s5` or `any` for any available slice). The remaining arguments are the size (from the following list), the file system to create, and optional parameters.

Table 6.4 *(Continued)*

KEYWORD	DESCRIPTION
	Size Values for **filesys** Keyword: number—For example, 500 existing—Whatever the current size is auto—Determined by software selection all—All the space on the disk free—All the available space on the disk start:size—Starting cylinder and number of cylinders File System Types: mount point name—For example, /apps swap—File system is to be used for swap space overlap—The entire disk unnamed—Raw partition ignore—Ignore the specified file system Optional Parameters: preserve—Preserve the existing file system on the specified slice mount options—For example, -o ro as with the mount command
geo	Designates the regional locale or locales that you want to install or add during an upgrade. These include N_Africa, S_America, C_Europe, S_Europe, C_America, Asia, E_Europe, W_Europe, N_America, Ausi, N_Europe, and M_East
install_type	Defines the type of installation. Possible values are initial_install, upgrade, and flash_install. An initial_install will recreate file systems and leave you with a rebuilt system. An upgrade will preserve most of the modifications you've made to the system and overwrite only those files that are required by the new OS. A flash_install will install from a Web Start Flash archive.
isa_bits	Determines whether 32- or 64-bit packages are installed. The default for UltraSPARC is 64-bit while the default for all other systems is 32-bit.

(continues)

Table 6.4 Profile Keywords *(Continued)*

KEYWORD	DESCRIPTION
`layout_constraint`	Specifies a constraint to be used when reallocation is required during the installation. If you do not use this keyword, all file systems requiring more space or affected by other file systems requiring more space (i.e., residing on the same disk) are considered changeable and may be rebuilt during the installation. The `backup_media` keyword will determine what space is used during this movement and reallocation. File systems identified with the `layout_constraint` command, on the other hand, will have the particular constraint enforced. These constraints include: `changeable`—Okay to move or to resize. `movable`—Can be moved, but not resized. `available`—Can be sacrificed if space is required. `collapse`—Can be merged into parent file system (i.e., file system where its mount directory is located) if necessary; this works only with file systems specified within the `/etc/vfstab` file (i.e., not automounted). `minimum_size_value`—The particular value specified is the minimum size that the file system must have following reallocation.
`locale`	Designates localization packages for support of non-English sites. Locale can be set to any of the following: `zh`—Chinese `fr`—French `de`—German `it`—Italian `ja`—Japanese `ko`—Korean `es`—Spanish `sw`—Swedish `zh_TW`—Taiwanese
`num_clients`	Specifies the number of diskless clients. This keyword can be used *only* when the `system_type` is server. It is used to allocate space for client partitions.

Table 6.4 *(Continued)*

KEYWORD	DESCRIPTION
package	Specifies that a particular package should be added or omitted during the installation, regardless of whether it is part of the selected software group. Options are add and delete. add is the default. When upgrading a system, all packages already on the system will be upgraded unless you specify delete.
partitioning	Defines how the disks are partitioned during installation. The options include the following: default—The installation software selects the disk and slice for each file system subject to constraints that you have specified with the filesys and layout_constraint keywords. existing—The installation software uses the existing file systems. All nonsystem file systems (i.e., file systems other than /, /usr, /usr/openwin, /opt, and /var) are preserved.
root_device	Specifies the system's root file system. This keyword can be used with initial installations and with upgrades. With upgrades, this keyword specifies the root file system to be upgraded.
system_type	Specifies the role the system will play. This keyword determines whether other keywords can be used (e.g., client_arch). Possible values are standalone, dataless, and server. Diskless systems are not installed with JumpStart, but with the add_client_whatever command.
usedisk	Specifies a disk that will be used during the installation. If you specify one disk with this keyword, you must specify all disks. Otherwise, the specified disks are the only ones that will be used. Keep in mind that you cannot use this keyword in the same profile as the dontuse keyword, so you must select your strategy to include or omit disks.

To check a profile, use the pfinstall command with a -d or -D option. This command provides verification that the new profile creates the configuration you think you've specified. It does this by showing you how the system would be created using the new profile without actually installing anything anywhere. It only works for initial installations (i.e., not with upgrades). Be sure to specify the -d or -D option; these ensure that you are testing the profile rather than installing software. The -D option instructs the software to use the disk of the system you are logged into to test the profile. -d instructs the software to use a disk configuration file instead. Use one of the formats shown here:

```
/usr/sbin/install.d/pfinstall -D [ -c Cdpath ] profile
/usr/sbin/install.d/pfinstall -d disk_config_file [ -c Cdpath ] profile
```

You need to run profile tests on a system that is running the same version of Solaris as the clients for which the profile is intended.

To add a rule to the rules file on the profile server that describes the client and identifies the profile that should be used during installation of clients of this type, the rules file (e.g., it could be called `/export/install/rules`) has the following format:

```
[!]keyword value [&& [!]keyword value] begin-script profile finish-
script
```

Here's an example of a rule:

```
karch sun4u - test-box.prof -
```

You need to have a validated rule in the `rules.ok` file, including a reference to a profile set up for this particular set of systems. You need to identify the systems and describe the target configuration.

A rules file has a rule for each distinct set of systems that you intend to install. It must reside in the JumpStart directory (for our purposes, we'll refer to this as `/export/jumpstart`). Profiles, on the other hand, can be used by more than one rule. This is helpful if you have several populations of systems that need to be set up the same way—that is, you can differentiate these from the rest with a single rule.

To validate your rules file, use the `check` command available with the install media. The check script will create the `rules.ok` file.

Creating a Profile Diskette

An alternative to the profile server is the profile diskette. If you have systems that are not on your network or that you prefer to install using profiles stored on a diskette, you can take this approach instead. For SPARC clients, creating a profile diskette involves preparing a diskette with a file system. This can be either a UFS or a pcfs file system, but we use a UFS file system in the following example. The JumpStart files are then copied to the diskette.

```
# fdformat -U
# newfs /vol/dev/aliases/floppy
# eject floppy          (then reinsert)
# volcheck
# cp /export/jumpstart/rules.ok /floppy/floppy0
# cp /export/jumpstart/test-box.prof /floppy/floppy0
```

For x86 clients, the process is a bit more complicated. The diskette must contain boot software as well as the JumpStart files. In the following example, we start by inserting a Solaris x86 boot diskette into the drive.

```
# dd if=/vol/dev/aliases/floppy0 of=/export/install/x86_boot
# eject floppy          (insert a blank)
# volcheck
# fdformat -d -U
# dd if=/export/install/x86_boot
# eject floppy          (then reinsert)
# volcheck
# cp /export/jumpstart/rules.ok /floppy/floppy0
# cp /export/jumpstart/test-box.prof /floppy/floppy0
```

Creating Rules

Rules allow you to specify which clients will use each of the profiles you've created. Each rule selects a single profile for a group of clients identified by one or more of the keywords listed in Table 6.5, as illustrated in Figure 6.6. In addition, once a matching rule is found, the rest of the file is not read. Each rule has the following form:

```
keyword value [&& keyword value] begin_script finish_script profile
```

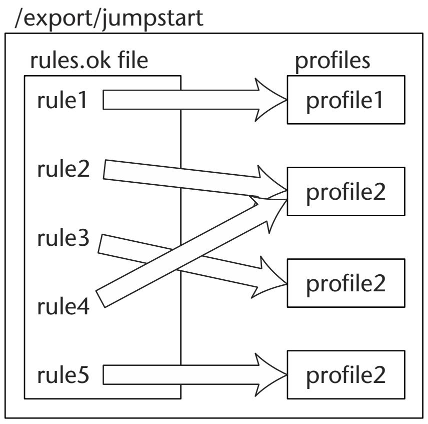


Figure 6.6 JumpStart rules and profiles.

Table 6.5 Rule Keywords

KEYWORD	DESCRIPTION	EXAMPLE
`any`	Matches any client	`any -`
`arch`	Specifies CPU architecture (SPARC or i386)	`arch SPARC`
`domainame`	Depends on client's existing domain	`domainname foo.com`
`disksize`	Specifies the client's disk size as a range (in megabytes)	`disksize c0t3d0 500-1000`
`hostaddress`	IP address	`hostaddress 10.4.3.111`
`hostname`	Specifies the client's host name	`hostname scotty`
`installed`	A disk with a root file system corresponding to the particular Solaris version	`installed c0t3d0s0 Solaris_7`
`karch`	Specifies the kernel architecture of the client	`karch sun4m`
`memsize`	Depends on the memory on the client (in megabytes)	`memsize 32-64`
`model`	Specifies the client model designator	`model SUNW, SPARCstation-20`
`network`	The system's network number	`network 10.4.3.0`
`osname`	Specifies the version of Solaris that is currently installed	`osname Solaris_8`
`probe`	Specifies a `probe` keyword to be used (returns information rather than identifying a match)	`probe disks`
`totaldisk`	Specifies a range of disk space (in megabytes)	`totaldisk 1000-2000`

A keyword can be preceded by an exclamation point (also referred to as a *not* sign) to effect its opposite. Any number of keyword/value pairs can be used if they are appended with double ampersands (also referred to as an *and* symbol). Begin and finish scripts are optional, but a hyphen must be used as a placeholder if there is no begin script.

A sample rule for a JumpStart installation on SPARC systems on the Test Group's subnet might look something like this:

```
arch SPARC && disksize 1000-2000 && network 192.9.201.0 - test-box.prof
-
```

NOTE This rule specifies the subnet and applies only to systems with 1 to 2 GB of disk space. It then specifies that the profile named `test-box.prof` is to be used to detail the rest of the installation parameters.

After the rules file is created (or any time you update it), you must run the check script to validate your rules and create the `rules.ok` file that will be used during installation.

```
# ./check
Checking validity of rules
Checking validity of test-box.prof
Checking validity of mgt.prof
Checking validity of ofc.prof
The auto-install configuration is ok.
```

Creating Begin and Finish Scripts

Begin and finish scripts are optional customization scripts that allow you to fine-tune your installations even further than your install profiles. Specific changes that you might otherwise make before or after installation can be automated and applied to any number of systems. Examples of things that you might find convenient to do in begin and end scripts include configuring network printers, customizing your windowing environment, installing patches, or adding `rc` scripts (discussed in Chapter 7) to automatically start site-specific services.

Begin and finish scripts are written in the Bourne shell (`/bin/sh`) and must be specified within the rules file. This means that for every client population you identify, you may identify a single begin and a single finish script. If you do not elect to use begin and finish scripts, you will need to put a hyphen in their positions in each rule.

All begin and finish scripts should be root-owned and have their permissions set to 644. Execute permission is not required, because these scripts will be provided as input to the `/bin/sh` command; this also lessens the chance that these scripts will be inadvertently run at other times.

File systems referenced in finish scripts should append `/a` to the beginning of any pathnames, because all the install client's file systems will be mounted within this mount point during and following the installation.

There are a number of environment variables (see Table 6.6) available within the installation environment that you can use within your begin and end scripts to facilitate your coding.

Table 6.6 Environment Variables for Begin and Finish Scripts

VARIABLE	DESCRIPTION
CHECK_INPUT	Specifies the path to the rules.ok file, mounted as /tmp/install_config/rules.ok
HOME	Home directory of root user during the installation
PATH	Search path during the installation
SI_ARCH	Hardware (CPU) architecture of the client, set by the arch keyword in the rules.ok file
SI_BEGIN	Name of the begin script, if specified; see SI_FINISH
SI_CLASS	Profile name
SI_CONFIG_DIR	Path of JumpStart directory
SI_CONFIG_FILE	Same as CHECK_INPUT?
SI_CONFIG_PROG	rules.ok file (without path)
SI_CUSTOM_PROBES_FILE	Used to define additional rule keywords
SI_DISKLIST	List of the client's disks
SI_DISKSIZES	Disk size of install client
SI_DOMAINNAME	Domain name, set when domainname keyword is used in the rules file
SI_FINISH	Name of the finish script
SI_HOSTADDRESS	IP address of the client
SI_HOSTID	Hardware address of the client
SI_HOSTNAME	Client's host name, if set in the rules file
SI_INSTALLED	Device name of disk with installed OS, set when installed keyword is used in the rules file
SI_INST_OS	Name of operating system
SI_INST_VER	Version of the operating system
SI_KARCH	Kernel architecture of the client, set in rules with karch keyword
SI_MEMSIZE	Amount of physical memory on the client, set with memsize keyword
SI_MODEL	Model name of the client, set with model keyword

Table 6.6 *(Continued)*

VARIABLE	DESCRIPTION
`SI_NETWORK`	Network number of the client, set with `network` keyword
`SI_NUMDISKS`	Number of disks on the client
`SI_OSNAME`	Release of OS on the CD or disk-based image
`SI_PROFILE`	Path to profile
`SI_ROOTDISK`	Device name of the rootdisk
`SI_ROOTDISKSIZE`	Size of the rootdisk
`SI_SYS_STATE`	The `/a/etc/.sysIDtool.state` file
`SI_TOTALDISK`	Total amount of disk space on the client
`SHELL`	Default shell for installation, `/sbin/sh`
`TERM`	Install client terminal type
`TZ`	Default time zone

Output from the begin and finish scripts can be examined later by looking at the log files—`/var/sadm/begin.log` and `/var/sadm/finish.log`—following the client's reboot.

Starting the Installation

To start a JumpStart installation over the network, once the servers are ready and the profiles and rules are in place, you merely have to enter the **boot** command on the halted client (i.e., at the ok prompt). Once you enter the appropriate boot command to boot over the network, (e.g., the following command), SPARC clients will begin the preinstallation sequence.

```
ok boot net - install
```

For x86 clients, you'll need to insert the Solaris boot diskette into the drive and then select a few options first.

Because very little has to be typed on the client with a well-crafted custom Jump-Start installation, the requirement for hands-on access to the client is limited. You could have a junior system administrator or a cooperative user start the installation for you, whereas they would probably be unable or too intimidated to perform a traditional installation. Figure 6.7 depicts booting over the network.

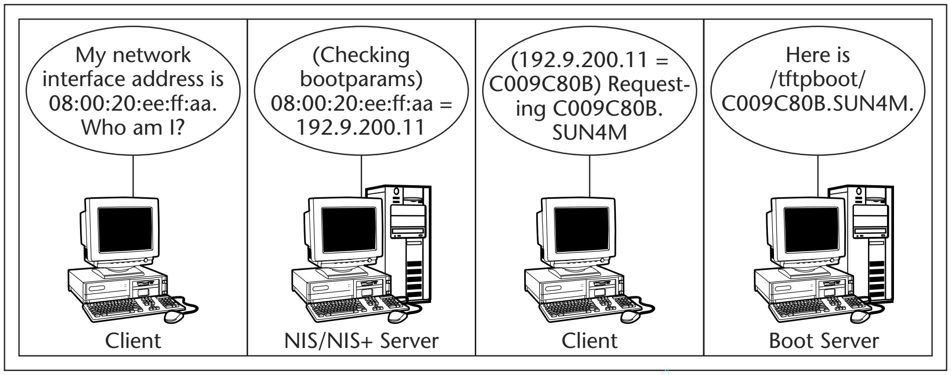

My network interface address is 08:00:20:ee:ff:aa. Who am I?
(Checking bootparams) 08:00:20:ee:ff:aa = 192.9.200.11
(192.9.200.11 = C009C80B) Requesting C009C80B. SUN4M
Here is /tftpboot/ C009C80B.SUN4M.
Client
NIS/NIS+ Server
Client
Boot Server

Figure 6.7 Booting over the network.

SPARC Clients

SPARC clients will start the preinstallation boot process by broadcasting a Reverse Address Resolution Protocol (RARP) request. Because the system Ethernet interface (hardware) address is stored on PROM, this information is available for use in determining the corresponding network (IP) address. After the client has received its IP address, it can use this information to request its boot program.

If you're using a profile diskette, the client will notice. Otherwise, it will look for a `rules.ok` file on the profile server. The client will obtain its IP address and then its boot file. The boot parameters will specify the profile server. The client will then mount its client partitions. If you have defined a begin script, it will be run before the installation begins.

If the installation client has a CD-ROM and you want to use it instead of a remote installation server to supply the installation media, you must supply the JumpStart files on a diskette. The installation software will find the `rules.ok` file on the diskette and, from it, determine the profile to use much in the same way it would if you were installing over the network (the first rule that matches the client). If none of the rules in the `rules.ok` file match the system, the software will initiate an interactive installation, abandoning JumpStart.

For systems that are not connected to a network, this is the only way you can use Custom JumpStart. It is possible, however, to prepare a diskette that can be used for a group of similar systems or to override the standard profile that a particular client would use if you were booting over the network.

Tasks That Need to Be Performed Once for Each Client

By using JumpStart properly, you will find that the work that needs to be done for each client being installed or upgraded is fairly minimal. Still, there are some steps that must be taken. These include the following:

- Preparing the boot server to provide boot support for this client
- Starting the installation

Just as with any installation, it is a good idea to back up systems before you reinstall or upgrade them. This is a precaution in case anything in the installation should fail, but is really needed only for files beyond the OS—configuration files and the like.

WARNING Some systems may not be able to run the newer OS. If you have older systems, you should check whether they are supported in the new release before making plans to reinstall or upgrade the OS.

Depending on the release of SunOS installed on a system, you may have the option to upgrade rather than install. This is likely to take less time, but, more importantly, an upgrade operation will not overwrite your configuration files.

If the system you're installing has a local CD-ROM that you want to use in place of the installation server, you will have to present the client with a profile diskette.

For Intel-based systems (not supported by Solaris 9 at this writing), insert the Solaris boot diskette into the local drive and hit reset or power cycle the system. At that point, you can choose to boot from a local CD or across the network. After selecting network boot, you have to select the Custom JumpStart method (versus the interactive install method in which you are prompted for all that is required).

For network use, Custom JumpStart requires that you have already provided for network recognition of the client to be installed. When it broadcasts a request for identifying information on being told to boot over the network during the initial phase of the installation, some server must be prepared to respond, and some server must have boot software that the client can use. For this to be the case, you must have provided information about the system and prepared a boot server. That is to say, the server must both be able to identify the client in question (i.e., provide an IP address corresponding to the hardware or Ethernet address of the client) and, subsequently, provide a bootfile and a root file system. Preparing a server for this role is fairly minimal work, provided it has adequate space (check to make sure this is still the case).

Add client support for each system to be installed by using the `add_install_client -c` command or by adding a wildcard entry to the `bootparams` file or map that allows any system to determine the location of the jumpstart directory (`* install_config=servername:/export/jumpstart`). This allows a single entry to provide a response for any client installation.

```
add_install_client -c profile-server:/export/jumpstart hostnamearch
```

It is useful to understand what the `add_install_client` command does, what files it creates, and how they are used. First, the command adds an entry to the `/etc/bootparams` file on the install server similar to the following:

```
client-name root=enterprise:/export/exec/kvm/sparc.Solaris_7
install=install-server:/export/install/Solaris_7 boottype=:in
install_config=profile_server:/export/jumpstart÷
```

This entry specifies, among other things, where the profile JumpStart directory is located. It also adds files to the `/tftpboot` directory on the boot server (or the `/rplboot` directory if this is an x86 system). It enables, if necessary, the `in.tftpd` daemon (see the /etc/inetd.conf file). It adds the client's hardware address to the ethers file and its name and IP address to /etc/hosts. The command also ensures that the host entry in the /etc/nsswitch.conf file includes a correct entry and that the path to the CD image is shared (it adds a line to /etc/dfs/dfstab). If necessary, it will also start certain daemons (e.g., rpc.bootparamd, mountd, nfsd, in.rarpd, and rpld,).

If the client is an x86 system, the add_install_client command sets up the /rplboot directory with the inetboot program for x86 clients, as shown here:

```
inetboot.i86pc.Solaris_7
```

The link itself has the client's IP address as part of the name (e.g., 192.9.200.11.inetboot), as shown in the listing below.

```
192.9.200.11.glue.com -> gluecode.com
192.9.200.11.hw.com -> smc.com
192.9.200.11.inetboot -> inetboot.i86pc.Solaris_7
gluecode.com
inetdboot.i86pc.Solaris_7
rm.192.9.200.11
smc.com
```

Summary

This chapter discusses using JumpStart on both Sun and x86 systems and suggests the sequence of steps that you need to take to get the most benefit from the tool. It provides some information on how to create and use Web Start Flash archives to further streamline Solaris installations. JumpStart can save you tremendous amounts of time and effort if you are responsible for installing or upgrading numerous systems. JumpStart can also be used for installation of patches.

JumpStart and Web Start Flash are particularly valuable if you:

- Are installing many systems with the same basic setup
- Want to make it possible for junior sysadmins or users to assist with the work of upgrading systems
- Are installing patches on numerous systems

Should you take the wise step of using JumpStart, you'll need to prepare your installation servers and a profile for each client type. The syntax is not difficult and, properly set up, reduces installations to little more than getting them started.

CHAPTER

7

Setting Up Name Services

This chapter details the installation and configuration of the Domain Name System (DNS). Chapter 8 discusses NIS, NIS+, and LDAP. In this chapter, we will provide a brief but sound picture of what the name service is and how it works. After that, we will first detail installation and configuration from the server side, and then work our way to the client configuration.

Even if you are not responsible for supporting DNS within your organization, an understanding of how DNS works and the extent to which the Internet would *not* work without DNS is essential to system administration. This chapter provides a gentle but fairly thorough introduction to the services, files, and options involved in running and using DNS.

NOTE Be aware that DNS setup changes significantly between Solaris 2.6 and Solaris 7. Because many of you are still running older versions in some shape or form, both configurations will be discussed in detail.

Domain Name System

The implementation of the Domain Name System or Domain Name Service (DNS) that ships with Solaris is called the *Berkeley Internet Name Daemon* (BIND).

Installing the DNS

Starting with Solaris 7, the version of BIND implemented is BIND version 8.1.x. Solaris 9 supports BIND 8.2.4. However, many companies are still running the older version, version 4.9.x. To determine the version of BIND you are running on your system, use the following command:

```
/usr/ccs/bin/what /usr/sbin/in.named | grep named
```

For more information on the latest version of BIND, you should refer to the Internet Software Consortium, which sponsors the development of BIND. Its Web site is located at www.isc.org/bind.html. In the sections that follow, we start by detailing the installation of the older version of BIND and then discuss the changes for the newer version.

TIP Carefully plan your DNS configuration before you begin to create it. Consider the natural boundaries of your organization as a template for where you should construct subdomains.

The steps involved in setting up DNS are quite logical:

1. Set up the boot file that denotes the DNS primary and secondary servers, and the zones for which they are authoritative. For versions of DNS preceding version 8, this is the file `/etc/named.boot`. For BIND version 8, this is the `named.conf` file.
2. Create the forward and reverse map files to detail the systems within each zone. These files are the same for all releases.
3. Start the `in.named` daemon to start serving DNS entries.

The Boot File

The two versions of the boot file, `/etc/named.boot` and `/etc/named.conf`, are detailed in the following sections. We also mention the script `named-bootconf`, which creates a file with the new format (i.e., a `named.conf` file) from one with the old format (i.e., a `named.boot` file).

named.boot

The first step in setting up DNS in older versions of Solaris (before BIND 8) is to create the `/etc/named.boot` file. This file details the zones and the servers that are authoritative for these zones. It also points to the `root.cache` file that knows how to reach all of the root name servers.

When the `in.named` daemon is started or sent a `sig-hup` (`kill -1`) signal, it reads the `named.boot` file. The server's start script is `/etc/init.d/inetsvc` (i.e.,

/etc/rc2.d/S72inetsvc). For each domain, the configuration file, named.boot, directs in.named where to get its data—from local data files if it is primary, and from other servers if it is a secondary.

The following listing shows a typical named.boot file:

```
directory /var/named
cache            named.ca
forwarders       10.69.0.1 10.69.1.1
primary          foo.com               foo.com.db
primary          0.0.127.in-addr.arpa  0.0.127.db
primary          2.224.in-addr.arpa    69.10.db
secondary        adamrachel.com        10.69.2.1   adamrachel.com.db
secondary        1.192.in-addr.arpa    10.69.2.1  1.192.db
```

The lines in the named.boot file have the following meanings:

directory. This line identifies the root path where named will look for all of the data files. If no directory is specified, named will look for files relative to the /etc directory. If a subdirectory is desired, list it without preceding slashes (e.g., named would represent /etc/named).

cache. This line identifies the file that named uses to get back to the root name servers for domains for which it is not authoritative and doesn't know the authoritative source. It contains the name and IP addresses of the root name servers on the Internet. Make sure your cache file is up to date, or else queries may not be answered for domains beyond your span of control. The latest version of the cache file can always be pulled from the following URL:

```
ftp://rs.internic.net/domain/named.root
```

NOTE **Don't confuse this cache file with the name server's internal cache (used to cache DNS responses).**

primary. Each of these lines specifies a zone for which this machine is authoritative. The first argument is either the domain suffix to be appended to every label in the file or, in the case of a reverse table, the reverse of the IP address that is to be prefixed to every network address in the file followed by in-addr.arpa. Reverse entries are a major source of confusion because the IP network 10.69 is coded as 69.10, and the 192.1 network as 1.192. The usual cause for failure of reverse lookups is the improper use of these lines. If your reverse name lookups don't work, check the addresses carefully.

forwarders. This is a list of addresses to which this machine should send forward requests for sites it cannot resolve. First, it will attempt to get an answer from the forwarders and, if that fails, from the root name servers. A good choice for a forwarder is the name server that is authoritative for the zone above yours.

secondary. A secondary line indicates that you intend this server to be a secondary name server for the listed zone. You should have at least one secondary server for each zone. Proper placement of secondaries can dramatically increase the reliability and performance of your DNS service. In a distributed organization, each area can be primary for its own subdomain and secondary for any other subdomains. Where this gets very tricky is in determining who is primary for the reverse tables. Because most companies utilize either single class B or C networks and physically co-locate people and machines in different logical subdomains, it may be a centralized network administrator who maintains the reverse maps. In this example, the secondary line will cause this machine to contact the name server at 10.69.2.1 and request a zone transfer for the `adamrachel.com` domain and the 192.1 subnet. You can specify multiple IP addresses to query on the secondary line. `named` will query those addresses for all the information it can get about the domain, remember it, and act as though it were authoritative for that domain.

named.conf

BIND 8 does not use a `named.boot` file but, instead, utilizes a `named.conf` file. You cannot have both of these files on your system at the same time. Sun ships a command called `named-bootconf` that can be used to automatically transform a `named.boot` file into a `named.conf` file. No translation is provided for moving backwards.

Although the `named.conf` contains similar information to `named.boot`, it is considerably more powerful, adding options that allow for the use of access control lists and categorized logging. The structure of the `named.conf` differs greatly from that of `named.boot`. Not only can you detail what zones a system is authoritative for, but there is a mechanism for restricting access to only those systems that you want using your name server. The `named.conf` file is split into sections. The major sections are as follows:

options. Sets default values and global settings.

include. Essentially the same as an include file for C; inserts the listed file at this point.

logging. Details what to log and where.

server. Sets options for a remote server.

zone. Defines a specific zone.

ACL. Specifies an access control list using dotted decimal or slash notation.

The following listing shows the `named.boot` file that was shown in the earlier listing expressed in the `named.conf` format:

```
options {
        directory         "/var/named";
        forwarders        {
                10.69.0.1;
```

```
                  10.69.1.1;
        };
            };
zone "." in {
        type hint;
        file "named.ca";
};
 zone "foo.com" in {
        type master;
        file "foo.com.db";
};
 zone "0.0.127.in-addr.arpa" in {
        type master;
        file "0.0.127.db";
};
 zone "2.224.in-addr.arpa" in {
        type master;
        file "69.10.db";
};
 zone "adamrachel.com" in {
        type slave;
        file "adamrachel.com.db";
        masters { 10.69.2.1; };
};
 zone "1.192.in-addr.arpa" in {
        type slave;
        file "1.192.db";
        masters { 10.69.2.1; };
};
```

The `/etc/named.conf` file, like `named.boot`, establishes the server as a master, slave, or cache-only name server. It also specifies the zones over which the server has authority and what files it should read to get its initial data.

The Options Section

The options section starts the `named.conf` file and defines options that are to be applied globally. Within the options section are the following directives:

directory. Specifies the directory where the DNS data files are located.

forwarders. Provides the list of machines to send forward requests to when the server does not have the data.

forward. If set, forces any queries to be sent to the forwarders first. If the forwarders do not have the answer, only then is the lookup performed locally.

notify. Causes messages to be sent to the slave servers when there is a change in a zone for which this server is authoritative. Changes are not pushed to the slaves. Rather, when the slave servers receive the notification, they will check to see if they need to perform a zone transfer. This statement can also be placed in the zone sections to override the default. The default is yes.

recursion. Allows the server to act as a proxy and attempt to answer when a DNS query requests recursion, even if it does not have the information. If recursion is set to off and the answer is unknown to the server, it will return a referral to the client.

check-names. Specifies whether names are to be checked for inclusion of illegal characters (such as underscores), and under what conditions. The `check-names` directive has two parameters. The first parameter states the source of names to be checked. The options are `master`, `slave`, and `response`; `master` causes names in the master zone files to be checked, `slave` checks names in the slave zone files, and `response` checks responses received back from queries. The second parameter is the level of severity (i.e., how to respond)—`warn`, `fail`, or `ignore`; `warn` will log invalid names but return them anyway; `fail` will also log invalid names, but will not return them, and `ignore` causes the server to play dumb and simply hand out the results. Unless you change the default behavior, in BIND 4.9.4 and later, a primary name server will refuse to load a zone with bad characters and a secondary will load but complain, while bad characters in responses will simply be ignored.

The next statements allow you to decide what systems in your network can query your name server, and which machines can perform zone transfers from your machine. If you are serving names out to the Internet community, we highly recommend that you do not allow zone transfers outside of your organization.

allow-query. Specifies machines from which you will allow queries to your name server.

allow-transfer. Specifies machines from which you will respond to zone transfer requests.

The Include Section

The include section allows an external file to be used in place of local content. The specified file can be thought of as being inserted in the `named.conf` file at the point of the include command:

```
include "/etc/named.zones-a-e"
```

`named` will stuff the `/etc/named.zones-a-e` file into the current file before processing it.

The Logging Section

The `logging` section details what to log and where. It is split into two parts—channels and categories.

The *channel phrase* or *channel statement* is used to link output methods, format options, and severity levels with a name that can then be used with the *category phrase* to select how to log different message classes.

All log output is directed to one or more channels. There is no limit to the number of channels you can define. Four are predefined: `default_syslog`, `default_debug`, `default_stderr`, and `null`. These cannot be redefined. The default definitions are shown in the following listing:

```
channel default_syslog {
    syslog daemon;        # send to syslog daemon
    severity info;        # send priority info and higher
};
channel default_debug {
    file "named.run";     # write to the file named.run in current
                          # working directory.
                          # Note: stderr used instead of "named.run"
                          # if server is started w the "- f" option.
    severity dynamic;     # log at the server's current debug level
};
channel default_stderr {  # writes to stderr
    file "<stderr>";      # illustrative only; there's currently
                          # no way of specifying an internal file
                          # descriptor in the configuration language.
    severity info;        # only send priority info and higher
};
channel null {
    null;                 # toss anything sent to this channel
};
```

Every channel phrase must state where the messages for that channel are logged—either to a file, a specific syslog facility, or not at all (i.e., ignored). The channel can also state the minimum severity level that will be logged.

The word `null` as a destination for a channel forces the channel to ignore all messages and to ignore any other options set for that channel.

The `file` clause can include limits on how many versions of the file to save and on how large the file can grow. The `size` option sets the maximum size of the file. Once the logfile exceeds the specified size, logging to that file ceases. We recommended that you do not limit the size of logfiles that may be needed to debug problems in the future. The default is no limit.

If you choose to use the `logfile` option, versions of files will be saved with a numeric suffix, starting with 0 and ending with *n*-1, where *n* is the number of versions you choose to keep. The maximum number that can be kept is 99.

The `syslog` clause takes a single parameter, the syslog facility. (See Chapter 10 for more information on the `syslog` service.) The important thing to remember is that by using the `syslog` clause, you are delegating the logging responsibility to your `syslog` service, and logging will be handled in the same fashion as other process logging is handled by `syslog`.

The `severity` clause, like the `priorities` clause, determines the message level at which messages will be logged. Only messages at that severity level or higher will be either written to a log file or sent to the `syslog` server.

Specific debug levels can also be set with the following syntax:

```
channel specific_debug_level {
    file "debug.txt";
    severity debug 2;
};
```

In this example, messages of severity 2 will be logged to the `debug.txt` file any time the server is running in debug mode, even if the global debug level is set higher.

If the `print-time` option is enabled, the date and time will be logged. The print-time option should not be used with `syslog`, because `syslog` already prints the date and time. If `print-category` is selected, the category of the message will be also logged.

NOTE Once a channel is defined, it cannot be redefined. However, there is no limit to the number of new channels you can create.

The `category` phrase splits logging into four default categories: `default`, `panic`, `packet`, and `eventlib`. If not specified directly, the `category` phrase defaults to the following:

```
logging {
   category default { default_syslog; default_debug; };
   category panic { default_syslog; default_stderr; };
   category packet { default_debug; };
   category eventlib { default_debug; };
};
```

As you can see, the `category` statement simply connects category names with the type of logging device that will be used. This can be either a file or a call to `syslog`. If more than one logging device is required, separate them with semicolons.

NOTE There can be only one logging statement used to define all of the channels and categories that will be used in the name space. Additional logging statements will simply generate warning messages, and their definitions will be otherwise ignored.

There are many categories to choose from. This allows you to send the important logs where they need to be and to prune out the logs that are not important. If you don't specify a default category, it defaults to { default_syslog; default_debug; }

As an example, let's say you want to log security events to a file, but you also want to keep the default logging behavior. You'd specify the following:

```
# channel my_secure_channel {
file "secure_file";
  severity info;};
category security { my_secure_channel; default_syslog; default_debug; };
```

To discard all messages in a category, specify the null channel, like this:

```
category lame-servers { null; };
category cname { null; };
```

The categories shown in Table 7.1 are available. Each selects a different type of message.

Table 7.1 Message Categories

CATEGORY	SELECTS MESSAGES
`default`	If nothing else is specified, this is what you get: { `default_syslog`; `default_debug`; };
`config`	High-level configuration file processing.
`parser`	Low-level configuration file processing.
`queries`	A short log message is generated for every query the server receives.
`lame servers`	Messages such as `"Lame server on...."`
`statistics`	Statistics.
`panic`	Server shutdowns. If the server has to shut itself down due to an internal problem, it will log the problem both in the `panic` category and in the problem's native category. If you do not define the `panic` category, the following definition is used: `category panic { default_syslog; default_stderr; };`
`update`	Dynamic updates.
`ncache`	Negative caching.
`xfer-in`	Zone transfers the server is receiving.
`xfer-out`	Zone transfers the server is sending.
`db`	All database operations.
`eventlib`	Debugging information from the event system. Only one channel may be specified for this category, and it must be a file channel. If you do not define the `eventlib` category, the following definition is used: `category eventlib { default_debug; };`

(continues)

Table 7.1 Message Categories *(Continued)*

CATEGORY	SELECTS MESSAGES
`packet`	Dumps of packets received and sent. Cannot be sent to `syslog`, only to files. If not defined, this defaults to `packet { default_debug; };`
`notify`	The notify protocol.
`cname`	Messages such as "...points to a CNAME."
`security`	Approved/unapproved requests.
`os`	Operating system problems.
`insist`	Internal consistency check failures.
`maintenance`	Periodic maintenance events.
`load`	Zone loading messages.
`response-checks`	Messages arising from invalid responses.

The Zone Sections

Each zone section corresponds to a specified line in the `named.boot` file. The basic zone entry contains the name of the zone; whether this server is the master or the slave for that zone; if slave, who the master is for that zone; and the file to read. The directives in the zone section are as follows:

master. The master copy of the data in a zone. This was called a *primary* in BIND 4.x.

slave. A slave zone is a replica of a master zone. The following parameter specifies one or more IP addresses that this machine is to contact to update its zone files. If a specific file name is listed, then the replica will be written to that file. This corresponds to a *secondary* in BIND 4.x.

stub. A stub zone only replicates the NS records of a master zone.

hint. The hint zone is solely used to find a root name server to repopulate the root cache file. This is the same thing as a *caching zone* in BIND 4.x.

As in the `named.boot` file, class IN follows, but may be omitted since it is the default class.

Many of the options allowed in the `zone` section are the same as the options in the options section. The options are:

check-names. See the description of `check-names` under *The Options Section*.

allow-query. See the description of `allow-query` under *The Options Section*.

allow-update. Specifies which hosts are allowed to submit dynamic DNS updates to the server. The default is to deny updates from all hosts.

allow-transfer. See the description of `allow-transfer` under *The Options Section.*

max-transfer-time-n. Specifies a time (in minutes) after which inbound zone transfers will be terminated.

notify. If this server is authoritative for a zone, it sends messages to the slave servers when there is a change. Changes are not pushed to the slaves; rather, when the slave server receives `notify`, the slaves will check to see if they need to perform a zone transfer. This statement can also be placed in the `zone` sections to override the defaults. The default is yes.

also-notify. Specifies additional machines to be notified of updates. If the `notify` option is not set, `also-notify` will be ignored.

The ACL Section

The ACL section defines who has access to a resource. The choices are `any`, `none`, `localhost`, `localnet`, and dotted decimal IP addresses. Note that an address match list's name must be defined with ACL before it can be used elsewhere. The following ACLs are built in:

any. Allows all hosts.

none. Denies all hosts.

localhost. Allows the IP addresses of all interfaces on the system.

localnet. Allows any host on a network for which the system has an interface.

The Server Statement

The `server` statement defines the characteristics to be associated with a remote name server. This can be quite useful if there is a remote name server giving bad answers to queries. By creating a server entry marking the server's answers as bogus, you can quickly eliminate problems by preventing further queries from being sent to it.

Two types of zone transfers are supported, `one-answer` and `many-answers`. `one-answer` is the default and is understood by all 4.9.x and 8.1.x servers. `many-answers` is supported only by BIND 8.1.

The Cache File

The cache file contains name server (NS) and address (A) records for the root name servers. The next section details what the different record types are and how they are used.

The following listing gives an example of a cache file:

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache . <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A    198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A    128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS   C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A    192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS   D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A    128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS   E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A    192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS   F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A    192.5.5.2241
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS   G.ROOT-SERVERS.NET.
```

```
G.ROOT-SERVERS.NET.      3600000      A    192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS   H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A    128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS   I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A    192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                        3600000      NS   J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A    198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                        3600000      NS   K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A    193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS   L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A    198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS   M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A    202.12.27.33
; End of File
```

You should always work with the latest cache file. The easiest way to do this is to periodically retrieve it using FTP from `rs.internic.net`; get the `named.ca` file from the domain subdirectory. If you have dig, an excellent available freeware tool, an interesting way to retrieve the file is as follows:

```
dig @ns.internic.net . ns > /var/named/named.ca
```

The Forward Map File

A *forward map file* (also referred to as a *zone file*). is used to resolve addresses for machine names in the given zone. The @ sign signifies the start of the definition for the zone. All records will contain `IN`, denoting an Internet record. Other types of records exist. However, they are rarely, if ever, used. The following listing shows an example forward map file:

```
; Authoritative data for foo.com
;
@             IN        SOA nameserver.foo.com. postmaster.foo.com. (
                        23         ; Serial
                        10800      ; Refresh 3 hours
                        3600       ; Retry 1 hour
                        3600000    ; Expire 1000 hours
                        86400 )    ; Minimum 24 hours
nameserver    IN        A          224.2.130.1
www           IN        A          224.2.130.2
mail1         IN        MX   100   224.2.130.3
mail2         IN        MX   200   224.2.130.4
IN            HINFO     PII350     Solaris 7
ftp           IN        CNAME      www
nameserver2   IN        A          224.2.130.5
East          IN        NS         nameserver2
East          IN        NS         nameserver
```

The SOA record

SOA is the *start of authority* record. It contains the information that other name servers querying this one will require, and will determine how the data will be handled by those name servers. Each primary file pointed to in `named.boot` should have one and only one SOA record.

The two fully qualified names listed after the SOA label are the primary name server for the domain, and where to send problems with the zone or domain via e-mail.

NOTE Because @ signs cannot be used in this field, the postmaster entry should be read as `postmaster@foo.com`. If your e-mail address contains a dot, such as `E.Marks@foo.com`, it would have to be escaped with a backslash, and the entry would read `E\.Marks.foo.com`.

Note that the zone for which this forward map file is authoritative is not listed. It is strictly located in the `named.boot` file in the line that refers to this filename. The SOA record has five magic numbers: `Serial`, `Refresh`, `Retry`, `Expire`, and `Minimum`.

Serial. The first magic number is `Serial`, which stands for *serial number.* `Serial` is used strictly to determine whether secondaries are up to date. Many people like to use the date with a two-digit extension. We find it much easier to start with the number 1 and increment every time a change is made to the file. If changed address information does not seem to be making it to the secondaries, chances are that you either forgot to update the serial number or forgot to restart `named`.

TIP Decide to use sequential numbers or a date+ scheme for your serial numbers, and use it consistently.

Refresh. The time (in seconds) interval within which the secondaries should contact the primaries to compare the serial number in the SOA; in the preceding sample, this is every 3 hours.

Retry. The interval within which the secondary should retry contacting an authoritative server if it did not respond at the refresh interval.

Expire. The amount of time that a secondary will hold information about a zone without successfully updating it or confirming that the data is up to date (i.e., if it loses contact with the primary). This number should be fairly large so that the loss of your primary name server will not affect your network. From experience, we can tell you that it is not unusual to point client machines at the secondaries and have no clients query the primary name server directly.

Minimum. The default time to live (TTL) for all records in a zone not containing explicit TTL values. In other words, this denotes how long the information will be considered good without needing to check with an authoritative server for changes.

Other Records

There are many types of records that can be used in the forward map file. In this example, we show the use of A, CNAME, MX, NS, and HINFO records.

A. The A record is an address record. This is the most straightforward of DNS records. It simply ties a name to an IP address.

CNAME. The CNAME (canonical name) record is for aliasing hostnames and always points to a name that is defined in an A record. A CNAME record cannot point to another CNAME record. It should be noted that a CNAME record can point to A records in other zones that the server is authoritative for. CNAME records should be used to make it easier for users to determine the role that a server plays. For example, if we have our primary Web server `www.foo.com` acting as an FTP server, we can alias `ftp.foo.com` to that box. People who wish to upload or download files will most likely try the name `ftp.foo.com` without even thinking. Providing the CNAME results in fewer user calls and a more productive FTP site; this is presumably good for everyone.

MX. The MX (mail exchange) record points to a Simple Mail Transfer Protocol (STMP) mail server. It must point to an existing A record. In the preceding example, there are two mail servers listed, `mail1` and `mail2`. The number after the MX is an arbitrary numeric cost factor. The entry with the lowest cost indicates that this site should be tried first. The greater the cost as compared to other MX records for the zone, the less chance there is of the site being chosen. Because `mail1` has a cost of 100 and `mail2` has a cost of 200, `mail1` is the preferred mail server. If it cannot be reached, clients will try `mail2`.

TIP Set up at least two MX records, maybe three, for your organization. This will make it possible for email to continue to arrive even if your primary mail server is down.

NS. The NS (name server) record simply delegates a branch of the current domain tree to another name server. In the preceding example, `nameserver2` will be authoritative for the `east.foo.com` subdomain. Why would you want to do this? If you have a large domain that can be broken into autonomous areas that can handle their own DNS information, it is much more practical to distribute the database and the workload. While you still should maintain a central area for the information that everyone needs to see, you should delegate the authority for the other areas of the company so that they can manage their own information. Remember, every domain that exists must have NS records associated with it. These NS records denote the name servers that are queried for information about that zone. For your zone to be recognized by systems outside your zone, the server delegating responsibility for the zone must have an NS record for your machine in your domain. For example, delegating the `East .foo.com` domain, we have the primary name server for `foo.com`, called `nameserver`. Within the `foo.com` map, we have an NS record for the machine authoritative for the `East.foo.com` domain. This machine is `nameserver2`. The second NS line shows that `nameserver` is also a secondary for the `East.foo.com` subdomain.

HINFO. The HINFO (host info) record, which is hardly ever used, is put in place to detail system types. The entry consists of two values separated by white space, the former being the hardware type, and the latter being the software version. Some sites use this field for other information, such as contact information or pager numbers for the system. In this case, we are simply stating that `www.foo .com` is a PII 350 running Solaris 7.0.

The Reverse Map File

Reverse maps provide information for reverse lookups in which the request contains the IP address and the response is the fully qualified domain name of the system in question. Looking at the sample reverse map file shown in the following listing:

```
130.2.224.in-addr.arpa
@               IN      SOA nameserver.foo.com.postmaster.foo.com. (
; Serial
10800           ; Refresh 3 hours
3600            ; Retry   1 hour
3600000         ; Expire  1000 hours
86400 )         ; Minimum 24 hours
;
1               IN      PTR         nameserver.foo.com.
2               IN      PTR         www.foo.com.
3               IN      PTR         mail1.foo.com.
4               IN      PTR         mail2.foo.com.
5               IN      PTR         Name server2.foo.com.
```

You are probably asking why the file looks like this. Although it may not seem obvious, once you have implemented your reverse map files, you will find them extremely easy to manage and maintain.

The reverse map file starts with the same SOA record that is found in the forward map file. The main difference is that the only type of record found in the reverse map file is the PTR record. PTR records are "pointer" records that connect each IP address listed to a hostname.

The sample reverse map is called `0.0.127.in-addr.arpa`. This file is simply the reverse map for the `localhost` entry. When any machine in the `foo.com` domain requests the name for the address 127.0.0.1, the name server will return `localhost.foo.com`. The format of the address is the reverse of a normal IP address. For example, IP address 1.2.3.4 would be found in the `4.3.2.in-addr.arpa` file (assuming it's a class C address) with a PTR record for the address 1. Also note that the PTR records must be fully qualified and end in a period.

Following is the sample reverse name lookup file for the 224.2.130 network. Note that there are no reverse entries for CNAMEs, and that there can be only one entry for each A record. Also, the network 224.2.130 is implicit for all hosts listed. If this were a class B zone, the host portion would have been 224.2, and the host entries would all have been 130.1 through 130.5.

NOTE It is important to note that the PTR record must point to a machine that can be found in DNS. Also, although reverse maps are not required, some protocols and applications may not allow your machine to communicate if a reverse entry for it cannot be located.

Starting Up the DNS Server

Before we get to the client side of the DNS configuration, let us mention the command that is used to start up the DNS service and the script that starts this service during normal booting. To start the DNS service, you can enter the following command:

```
/usr/sbin/in.named &
```

The `in.named` process will read its configuration file (`/etc/named.boot` or `/etc/named.conf`) and start up in the background. Under normal conditions, you won't have to start this process manually. If the boot file and the `in.named` process file both exist on your system, the DNS service will start automatically when you boot a system. It is started in the `/etc/rc2.d/S72inetsvc` script.

TIP As arguably the most prevalent service running on the internet, BIND is often a logical target for hackers. When running DNS, it is imperative that you keep up with the latest security patches as found on the CERT (www.cert.org) and Sunsolve (http://sunsolve.sun.com) Web sites.

Setting Up Solaris as a DNS Client

To set up your Solaris system as a DNS client, you need to know two important pieces of information: the IP addresses of your name servers and what domain you are in (not to be confused with your NIS or NIS+ domain). There are only two files that need to be edited. The first is the `/etc/resolv.conf` file, and the second is the `/etc/nsswitch.conf` file. Here is an example of a simple `resolv.conf`:

```
domain foo.com
nameserver 224.2.130.2
nameserver 224.2.133.171
```

This file simply states the domain we are in and what name servers (in order) are to be queried for DNS information.

To update the `/etc/nsswitch.conf` file for DNS, simply add **dns** to the hosts line within the file. Determine the order in which you want the system to check name servers. In most cases, the order will be files and then DNS. If you are running NIS or NIS+, you may want DNS last, as shown in the following line:

```
hosts:  files nis dns
```

Troubleshooting Your DNS Environment

The first thing to do in troubleshooting your name server environment is to make sure that your client is configured correctly. If you are using a Solaris client, make sure that the `/etc/resolv.conf` has the correct domain and name server entries. Then, use either `nslookup` or `dig` (as discussed in the following sections) to try and find your problem.

TIP **Use both `nslookup` and `dig` to routinely check the functioning of your name server. Familiarizing yourself with the output when things are working well will help you to spot problems when they are not.**

Troubleshooting with nslookup

Next, type **nslookup** at a command prompt. If `nslookup` connects to your name server (as shown in the following example), you have at least verified that named is up and running. If `nslookup` cannot connect, make sure that `in.named` is running, and, as should be routine, check for messages pertaining to named in your `/var/adm/messages` file.

```
% nslookup
Default server:  ns.foo.com
Address:  224.2.130.1
>
```

Within the `nslookup` program, try a few simple queries such as the name of the name server and `localhost`. Both of these should resolve without a problem.

If you are connected to a slave, and the answers are not correct, try connecting to the master and checking again. If the master and slave differ, chances are that you forgot to update the serial number in the SOA in one of your maps, and a zone transfer did not occur when needed. Update the serial number on the master, kill and restart `named`, and check again. Remember, this information takes time to propagate.

If you can resolve forward but not reverse references, check your PTR records and make sure you have addresses in the reverse order. Also make sure that all of the entries end in `in-addr.arpa`.

Many resolvers will attempt to locate a host by traversing up the domain tree with the hostname. For example, if you are in the domain `a.b.c.com` trying to access a machine called `www`, the resolver will try all of the following names looking for this system:

```
www.a.b.c.com.

www.b.c.com.

www.c.com

www.com

www
```

If there are multiple machines with the name `www` at different levels in the organization, the resolver might provide the address of a different system than the one you want. Be sure you look at the result you are receiving, and not just that you are receiving a result.

Troublshooting with dig

If you are tired of `nslookup` and would like to use a more revealing tool, try `dig`. This freeware tool may be found on the Web and provides much more detail than `nslookup`. It is included with Solaris 9 and is part of newer BIND distributions.

To start `dig`, use the following command:

```
% dig host-name
```

Using `dig` in this manner causes dig to return the IP addresses (if any) for the given host or domain name along with a variety of additional information, as shown here:

```
dig www.foo.com
; <<>> DiG 2.0 <<>> www.foo.com
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
;; flags: qr rd ra ; Ques: 1, Ans: 2, Auth: 3, Addit: 3
;; QUESTIONS:
;;      www.foo.com, type = A, class = IN
;; ANSWERS:
www.foo.com.    85603   CNAME   evan.foo.com. <- Hey,it has another name!
evan.foo.com.   85603   A       224.2.130.2  <- And here's the address.
;; AUTHORITY RECORDS:                      <- Authoritative name servers
foo.com.        86257   NS      ns1.yourdomain.com.
foo.com.        86257   NS      ns2.yourdomain.com.
foo.com.        86257   NS      equinox.stockernet.com.
;; ADDITIONAL RECORDS: <- Addresses of those name servers
ns1.yourdomain.com. 83527               A       198.69.10.27
ns2.yourdomain.com. 83527               A       198.69.10.5
equinox.stockernet.com.   164809        A       199.170.68.11
;; Sent 1 pkts, answer found in time: 0 msec
;; FROM: workstation to SERVER: default -- 10.69.224.253
;; WHEN: Wed Jan 27 15:14:29 2002
;; MSG SIZE  sent: 37  rcvd: 209
```

You can see from this listing that `www.foo.com` has another hostname, identified by the CNAME record. That name is `evan.foo.com`. The IP address associated with both of these names is 224.2.130.2.

We can also see that there are three name servers that are authoritative for this record. Each is listed. The IP addresses for the three authoritative name servers are listed under the ADDITIONAL RECORDS heading.

There are a number of options available with `dig`. They are listed in Table 7.2 and are more fully explained in the examples that follow.

```
% dig -x ip address
```

The -x option of `dig` is used when you are looking for DNS information using an IP address instead of the usual domain name.

Table 7.2 dig Utility Options

OPTION	PURPOSE
-x	Uses IP address instead of FQDN
@nameserver	Uses a specific name server
axfr	Performs a zone transfer
any	Checks for any DNS information

For example, the command `dig -x 224.2.130.2` will return the name(s) assigned to the IP address 224.2.130.2, as shown here:

```
% dig -x 224.2.130.2
; <<>> DiG 2.0 <<>> -x
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
;; flags: qr aa rd ra ; Ques: 1, Ans: 1, Auth: 1, Addit: 1
;; QUESTIONS:
;;      2.130.2.224.in-addr.arpa, type = ANY, class = IN
;; ANSWERS:
2.130.2.224.in-addr.arpa.    86400   PTR     c235786-
a.blfld1.ct.spatulacity.com.               <- Machine name
;; AUTHORITY RECORDS:
2.224.in-addr.arpa.    86400   NS      dns1.blfld1.ct.spatulacity.com.
                                       <- Who's authoritative
;; ADDITIONAL RECORDS:
dns1.blfld1.ct.spatulacity.com.      86400   A       224.2.128.33
;; Sent 1 pkts, answer found in time: 0 msec
;; FROM: c235786-a to SERVER: default  224.2.128.33
;; WHEN: Wed Jan 27 15:05:50 1999
;; MSG SIZE sent: 45 rcvd: 125
```

To query a specific name server, simply add the @*ns* option, replacing *ns* with the name server's name or IP address This is also useful for determining if a change in a DNS entry has been propagated. Here's an example:

```
% dig @ns1.spatulacity.com www.foo.com
; <<>> DiG 2.0 <<>> @ns1.spatulacity.com www.foo.com
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 10
;; flags: qr rd ra ; Ques: 1, Ans: 2, Auth: 3, Addit: 3
;; QUESTIONS:
;;      www.foo.com, type = A, class = IN
;; ANSWERS:
www.foo.com.    86391   CNAME   evan.foo.com.
evan.foo.com.   86391   A       224.2.130.2
;; AUTHORITY RECORDS:
foo.com.        78217   NS      NS0.YOURDOMAIN.NET.
foo.com.        78217   NS      NS2.YOURDOMAIN.NET.
foo.com.        78217   NS      EQUINOX.STOCKERNET.com.
;; ADDITIONAL RECORDS:
NS0.YOURDOMAIN.NET. 168993  A       198.69.10.2
NS2.YOURDOMAIN.NET. 168993  A       198.69.10.5
EQUINOX.STOCKERNET.com.   168993  A       199.170.68.11
;; Sent 1 pkts, answer found in time: 3 msec
;; FROM: sweeney to SERVER: ns1.spatulacity.com 224.0.0.27
;; WHEN: Wed Jan 27 15:12:25 1999
;; MSG SIZE  sent: 37  rcvd: 212
```

Note that the results are the same as those generated without the @ option. This is normally the case unless a DNS change has happened recently.

Adding the string *axfr* to the command line will cause dig to perform a zone transfer for the given domain. Zone transfers are useful to check if a name server is properly set up to host a domain name and to track down various DNS problems. For example:

```
% dig foo.com axfr @ns1.yourdomain.com
; <<>> DiG 2.0 <<>> foo.com axfr @ns1.yourdomain.com
;; ->>HEADER<<- opcode: QUERY , status: REFUSED, id: 10 <- Note that
this request was refused.
;; QUESTIONS:
;;      foo.com, type = AXFR, class = IN <- No answers in this section,
either
;; FROM: sweeney to SERVER: ns1.yourdomain.com  198.69.10.27
;; WHEN: Wed Jan 27 15:18:00 1999
```

The zone transfer was refused. Let's try another one.

```
% dig axfr blfld1.ct.spatulacity.com @ns1.spatulacity.com
; <<>> DiG 2.0 <<>> axfr blfld1.ct.spatulacity.com @ns1.spatulacity.com
;; QUESTIONS:
;;      blfld1.ct.spatulacity.com, type = AXFR, class = IN
spatulacity.com.        95226   NS      NS1.HOME.NET.
spatulacity.com.        95226   NS      NS2.HOME.NET.
NS1.HOME.NET.    169147  A       224.0.0.27
     NS2.HOME.NET.  169147 A   224.2.0.27
```

In the preceding example, our request for a zone transfer was accepted. Normally, there would be quite a bit of output from such an operation. The domain in the example is small on purpose.

Adding the string *any* to the command line will cause dig to check for any DNS information rather than just the IP address:

```
% dig any www.foo.com
; <<>> DiG 2.0 <<>> any foo.com
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
;; flags: qr rd ra ; Ques: 1, Ans: 3, Auth: 3, Addit: 3
;; QUESTIONS:
;;      foo.com, type = ANY, class = IN
;; ANSWERS:
foo.com.         77302   NS      ns1.yourdomain.com.
foo.com.         77302   NS      ns2.yourdomain.com.
foo.com.         77302   NS      equinox.stockernet.com.
;; AUTHORITY RECORDS:
foo.com.        77302   NS      ns1.yourdomain.com.
foo.com.        77302   NS      ns2.yourdomain.com.
foo.com.        77302   NS      equinox.stockernet.com.
;; ADDITIONAL RECORDS:
ns1.yourdomain.com. 78758   A       198.69.10.27
```

```
ns2.yourdomain.com. 82924   A       198.69.10.5
equinox.stockernet.com.   164206  A       199.170.68.11
;; Sent 1 pkts, answer found in time: 0 msec
;; FROM: sweeney to SERVER: default -- 10.69.224.253
;; WHEN: Wed Jan 27 15:24:32 1999
;; MSG SIZE  sent: 33  rcvd: 200
```

We can see that quite a bit more information is returned with "any" specified.

Summary

DNS is one of the most critical networking components for any organization working on the Internet. Regardless of whether you are directly responsible for its maintenance, knowing how it works and how it is configured will help you in your job:

- Name servers either own information about your systems (in which case they are said to be authoritative) or they know how to get it from other name servers. This is true of the Internet at large, but can also be true within your organization—if you choose to distribute responsibility for your name space across the organization.
- Setting up a DNS server requires that a number of files be carefully, if not painstakingly, prepared. These files describe your systems, identify your name servers, and provide a means for ensuring that updates are properly propagated.
- DNS provides name-to-address as well as address-to-name translations. It identifies mail exchangers and those servers that are authoritative for the given domain.
- Setting up a client to use a name server is a much simpler operation than setting up a DNS server. There are only two files involved, and the entries are very simple.
- Should you have any trouble setting up your name server or ensuring that it is working correctly, there are two tools that you should be prepared to use—`nslookup` and/or `dig`. Sample commands and files included in this chapter should help you use these tools and become proficient at setting up, using, and troubleshooting DNS.

CHAPTER

8

Network Information Services

Something dramatic has happened with respect to Solaris information services since the first edition of this book. Though Solaris continues to provide support for all of the four name services—NIS, NIS+, /etc files, and DNS—that it has supported in the past, a new service has been added that will dramatically affect the way systems are managed in the future. Included with Solaris 9 is an LDAP (Lightweight Directory Access Protocol) service and all the support required to take advantage of it.

When the first edition of this book was printed, NIS+ appeared to be the best choice for distributing network information across a Solaris network. More secure and considerably faster for update operations than NIS, NIS+ seemed to provide what we saw as lacking in NIS, though at a considerable cost in terms of complexity. The focus of this chapter was, therefore, on NIS+. We talk along very different lines in this second edition.

Considering the growing presence of LDAP—an industry standard for distributing information across a network, and given Solaris and Sun's clearly stated commitment to LDAP, the direction in which we would choose to move today is unequivocal. Whether we make the move to LDAP in Solaris 9 or delay for another year or two, LDAP is the direction in which most of us will be moving in the not too distant future, and it is therefore the major focus of this chapter.

Solaris Name Services

All of the following name services are provided in Solaris 9:

- `/etc` files, the original Unix configuration files for users, hosts, and so on. Since this simple naming service is common to all versions of Unix, they won't be discussed further here.
- NIS (formerly `yp`), the Network Information Service, which provides this information through "maps" available for sharing across a network.
- NIS+, a hierarchical and more secure service developed as a replacement for NIS.
- DNS, the Domain Name Service, used to resolve hostnames across the Internet. See Chapter 7 for more information on DNS.
- LDAP, a general-purpose directory service with available schemas to serve data supplied by NIS, NIS+, or DNS.

Before we get into the details of how LDAP works, let's review the basic differences between these network information services:

- LDAP can be used to distribute almost any type of information across an organization, going far beyond the traditional substance of network information systems such as login and system information. It might be used, for example, to distribute employee telephone extensions and cubicle numbers, email addresses, or information about projects and customer contacts. LDAP, used as a replacement for NIS or NIS+, will provide the same types of data via a different protocol and storage mechanism. The primary advantage of LDAP is that it is or soon will be ubiquitous. Every major vendor supports LDAP in one way or another. This is in sharp contrast with NIS+, which is primarily available only on Solaris.
- The NIS name space is a *flat* namespace, which means that it does not support subdomains. Under NIS, only one domain is accessible from a given host. In both LDAP and NIS+, the namespace is *hierarchical.* For any change in NIS, the entire map must be propogated to slave servers.
- NIS+ also has much improved security over NIS. NIS does not support authentication, authorization, or secure remote procedure calling (RPC), whereas NIS+ supports authentication, authorization, and secure RPC. LDAP provides sophisticated authentication and authorization components.

The use of any of these services requires proper configuration of the name service switch file, `/etc/nsswitch.conf`, which directs applications to various sources of naming information. Figure 8.1 illustrates the relationship between applications, the calls they make for naming information (such as `gethostbyname`), the service switch, and the underlying naming service.

In specifying services within the /etc/nsswitch.conf file, you can now use files, nis, nisplus, dns, and ldap to refer to any of these services.

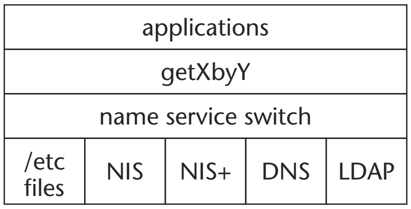


Figure 8.1 Solaris name service switch.

Though Solaris will continue to support all of these services through the life cycle of Solaris 9, it is unclear whether NIS or NIS+ will remain viable long afterwards. Sun has warned that NIS+ is likely to be dropped after Solaris 9. Sun's clear direction and the direction of the industry as a whole appears to be LDAP. Let's take a look at exactly what LDAP is and how it's likely to affect your network and your job in the not too distant future.

Lightweight Directory Access Protocol (LDAP)

The Lightweight Directory Access Protocol (LDAP), pronounced "el-dap", was first included in Solaris with the Solaris 8 release, although one might describe its presence in that release as dormant. In other words, it was there for developers to tinker with, but none of the system services relied on it in any way. The situation is different in Solaris 9, where LDAP support has been integrated into everyday tools. In other words, these tools are as ready to bind to an LDAP server as to local files or other services for their data. In addition, an LDAP server, the Sun ONE Directory Server, version 5.1 the (e.g. the iPlanet Directory Server), ships on the companion CD with Solaris 9 and can be used to set up and configure an LDAP server to provide support to systems transitioned into LDAP clients.

What Is LDAP?

First and foremost, LDAP is a protocol. It defines a set of operations that provide a way for information to be shared across a disparate set of systems and applications. Like many Internet protocols, LDAP operations involve servers and clients. LDAP servers maintain information along with details of how that information is organized. Clients request information for processing requirements, such as determining the home directory of a user who is logging into a system. LDAP was originally created by the University of Michigan in the early 1990s but has gone through quite a bit of evolution since then.

LDAP stands for *Lightweight Directory Access Protocol*. The word *Lightweight* refers to the fact that the protocol is efficient, far simpler and easier to use than its predecessor

DAP (Directory Access Protocol), which was associated with the development of X.500. Like its X.500 ancestor, LDAP provides an access protocol along with a set of naming rules that allow data to be easily located.

NOTE The word *directory* should be construed as meaning a *lookup service* in the same vein that a telephone directory is a *directory*. The reference has nothing to do with file system directories.

Though LDAP is rightly viewed as a replacement for NIS, for NIS+, or even for `/etc` files, its significance is far more extensive than these services. LDAP is a general-purpose tool that can provide lookup services on diverse platforms, such as Solaris and Windows. LDAP also shows tremendous promise as a lookup service for use with Web sites. LDAP lookup forms for email addresses are already appearing on some Web-based mail systems.

Though LDAP is often talked about as though it were a database, it is not a database in the manner that Oracle and Sybase are databases. At the same time, it does provide mechanisms with which information can be stored, modified, and retrieved. In general, LDAP operations are faster than database operations because the protocol is optimized for reads and much less efficient for updates—in keeping with the expected behavior of LDAP services.

Based on a set of RFCs and under the control of the IETF, LDAP provides a data sharing mechanism that is independent of operating system, underlying hardware, and application peculiarities—and unlikely ever to be dominated by a single vendor. Where sharing information about a set of users in a mixed environment involving Solaris, Windows, Linux, and MacOS systems would have been nearly if not specifically impossible in the past, supporting such a mixed environment using LDAP is not only entirely possible, it is the direction in which the technology is intended to take us.

Though similar to NIS+ in both sophistication and organization, LDAP offers numerous advantages including:

- Greater scalability with support for millions of records
- Greater flexibility with its ability to distribute dissimilar information across a group of servers
- Greater utility through its use by virtually any application
- Greater availability with the addition of replication services
- Greater security

LDAP is essentially an object-oriented service. This implies a particular organization and a structured relationship between its various records. In a broad sense, LDAP directories make it possible for system administrators to name, manage, and access collections of records.

The overall LDAP information space is, like a Unix file system, organized in a tree-like fashion. Because LDAP is organized hierarchically, it makes sense to begin our discussion of LDAP components with the root-level element.

The overall structure is called the *LDAP directory tree*. The root of this structure is referred to as the *base distinguished name*, or *base DN*.

Every record in an LDAP directory has a unique distinguished name (DN). In similar manner to a fully qualified domain name (FQDN) in DNS, such as `www.dragonflyditch.com`, a DN includes a string associated with the specific record following by names related to each higher-level component, ending with the base DN. These and other terms associated with LDAP are defined in Table 8.1.

If you were using LDAP to store information about all of your personal possessions, for example, you might have a record that refers to a pair of purple socks. The location of these socks might be the second drawer in the dresser in your bedroom in your home. Your LDAP records would reflect these locations hierarchically.

Each DN is composed of two components:

- The *relative distinguished name*, or RDN—The RDN is similar to a Unix file name.
- The *base distinguished name*—The location within the LDAP directory where the record resides. This resembles the full pathname where the file is stored.

The RDN of the socks record would reflect the description "purple socks," while the location would be drawer2, dresser, bedroom, home. There would be a clear distinction between the socks and the location of the socks. The socks are a possession, while the hierarchy of elements pointing to the location of these socks are something else altogether (let's ignore the fact that you might also own the dresser along with the room and the house). If you had two pair of purple socks, the second pair would have a different name (e.g., purple socks pair #2), or you'd have to store them in a different drawer or make sure one pair is always in the laundry.

Each DN uniquely identifies an object with the information space of an LDAP server in a manner similar to the way that a fully qualified domain name identifies a unique host on the Internet. For example, the root of an LDAP tree might be expressed like this:

```
dc=dragonflyditch, dc=com
```

Table 8.1 LDAP Terminology

ACRONYM	MEANING
LDAP	Lightweight Directory Access Protocol
DN	Distinguished name
RDN	Relative distinguished name
DIT	Directory information tree
LDIF	LDAP Data Interchange Format
OID	Object identifier

There are two naming standards that are used to define the names of LDAP trees. One, used in the example above, involves breaking the name into its domain components (“dc” refers to a domain component). The other is to use a single parameter to refer to the organization. In this way of naming, the following record would be used instead:

```
o=dragonflyditch.com
```

Of these two, the first seems to be the most popular.

Why Use LDAP?

LDAP was designed to promote interoperability among systems of various types and has gained enough support in the past couple of years to have finally achieved critical mass. We now believe that it is inevitable that LDAP will become the standard tool for distributing network information and data of many kinds within organizations and on the Internet.

Not surprisingly, nearly every modern language now includes an LDAP API and, because the protocol is simple, the similarity from one to the next suggests that system administrators and programmers will come quickly up to speed on any of them.

In addition, many different LDAP servers are available, both free and commercial. While the inclusion of the Sun ONE Directory Server (formerly iPlanet Directory Server) in Solaris 9 will undoubtedly make it the tool of choice for Solaris servers, the easy availability and configuration of servers for all platforms will promote the quick acceptance of LDAP for many types of corporate-wide information.

Most, if not all, LDAP servers include some type of replication technology providing higher availability. While, the transition may be slow, the direction is clear.

Its hierarchical nature may be less flexible than relational stores, but generally reflects the nature of information within an organization. The hierarchical nature of LDAP also facilitates the delegation of responsibility for portions of the overall directory structure.

In short, LDAP is:

- Multiplatform, unlike NIS+
- Fast and secure, unlike NIS
- Distributed by design, unlike `/etc` files

Not only does LDAP overcome the limitations of NIS (poor security and limited extensibility) and NIS+ (which is not supported on non-Solaris platforms), but LDAP is well integrated into Solaris 9 and clearly the chosen migration strategy of Sun itself.

LDAP also provides specific advantages that suggest its use. Because the protocol provides one of the least expensive (the software is free and included in Solaris 8 and above) and most universal ways to share information across an organization regardless of the types of systems in use, it lends itself to any type of information that is widely used in a company. It is most effectively used when:

- Information is infrequently changed (e.g., hopefully, information about your employees).
- Information is required by numerous applications.
- Information must be available to different types of systems.
- Portions of the information would be best managed by individual units.

In short, LDAP also allows organizations to distribute many types of information that the designers of NIS and NIS+ never considered. It is fully extensible and does not require a single master server to synchronize the information that it distributes. LDAP is clearly the "technically correct" way to distribute information on today's networks.

LDAP Basics

Although you may well have heard about LDAP, the underlying structure may be unfamiliar. In the following sections, we introduce some of the basic concepts and structural elements.

The LDAP Schema

An LDAP schema is similar in purpose to a database schema. It defines the contents—fields and types—associated with stored records. Following your transition to LDAP, for example, the data in your current naming service will be stored in accordance with the appropriate schema. For example, the schema for a POSIX account has the following, mostly familiar, fields:

cn (commonName). The username.

gidNumber. The group number.

homePhone. Individual's phone number.

uid (userID). The username again.

uidNumber. The numeric user ID.

homedirectory. User's home directory.

gecos. Usually, the full name of the user.

loginShell. The user's login shell (`/bin/bash`).

userPassword. The password field (as stored in the `/etc/shadow` file).

LDIF

The format used to move information out of and into LDAP is called *LDIF (LDAP Data Interchange Format)* and provides a simple tagged format that is easy to understand. LDIF uses a simple text format. A typical record might look like this:

```
dn: cn=Skooter Daniels, o=Foo Ltd, c=US
cn: Skooter Daniels
cn: Skooter
sn: skood
mail: skood@foo.com
objectClass: top
objectClass: person
```

This format describes a directory and directory entries in clear fashion. LDIF is commonly used to initially build a directory database as well as to add large numbers of entries in a single step. The record format also provides a way to describe changes to directory entries. Most of the directory server's command-line utilities use the LDIF format for input or output.

TIP The simple text format also makes it relatively easy for a system administrator to create records using their favorite scripting language.

The primary database used by an LDAP server is internal, while the LDIF is easy to read and only approximates the syntax used internally. This format provides data conversion to and from LDAP servers for both adding and extracting data and for moving data between LDAP servers with different internal formats.

The basic format for an LDIF entry looks like this:

```
[id]
dn: distinguished name
objectClass: object_class
objectClass: object_class
...
attribute_type:attribute_value
attribute_type:attribute_value
...
```

In this format, *id* is an optional numeric identifier created by LDAP. The `dn` entry is the distinguished name. The objectClass variable identifies the record type to be used. The attribute types and values hold descriptive information about the object.

DIT Layout

Solaris naming information, when stored in LDAP, makes use of a series of record structures, or *containers*, that allow the specific information needed; for example, for logins and identification of protocols to be made available through the new service.

LDAP Operations

An LDAP session starts with a BIND operation, much like NIS or NIS+. Although the service is based on a client/server architecture, the name space can be partitioned in many ways. In fact, LDAP provides load balancing as well as data partitioning. Figure 8.2 illustrates a typical LDAP structure in which three servers are each responsible for a portion of the overall information space. Each circle in the figure represents a different record in the overall tree. Multiple directory structures are linked by something called *referrals*. These are instructions from LDAP servers to LDAP clients requesting the clients to continue their search on another directory server—not unlike a redirect on a Web site.

LDAP Commands

A Solaris system that has been configured as an LDAP client is called a *native LDAP client*. Clients that are not yet native LDAP clients can take advantage of LDAP API tools provided to ease the transition to LDAP.

Native LDAP commands include `ldapclient` and `idsconfig`.

The `ldapclient` command initializes a system as an LDAP client. It can also be used to restore the service for a client and to display the LDAP client cache.

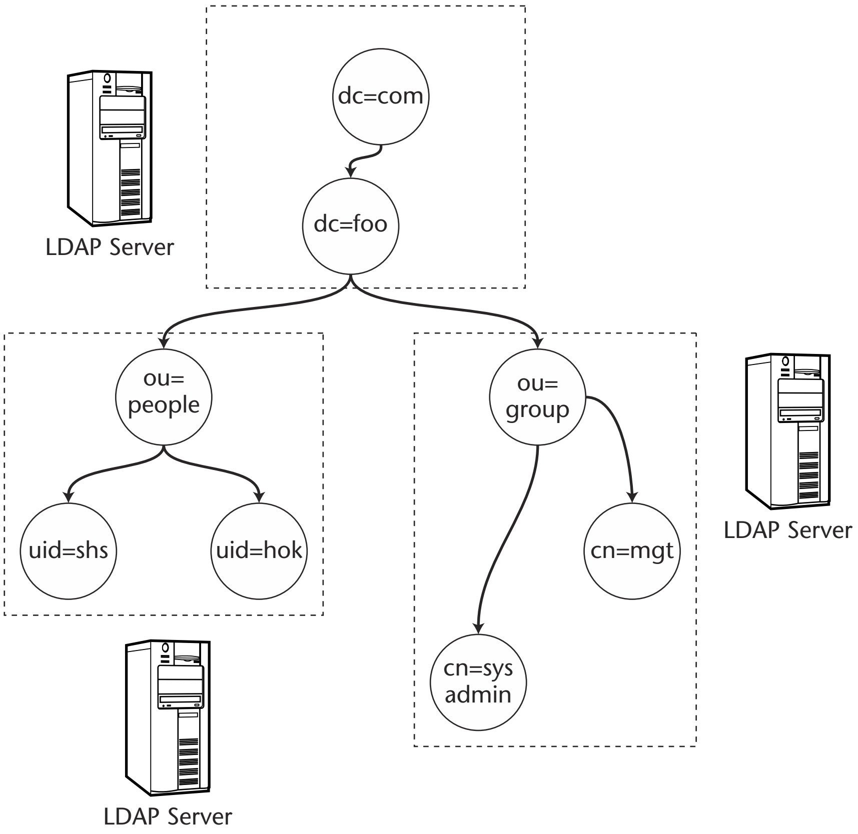


Figure 8.2 LDAP servers.

The `idsconfig` command prepares the Sun ONE Directory Server to be populated with data so that it can support the clients. The tool can prompt the user to supply data or data can be fed to the command on the command line by using the -i parameter. The syntax of the `idsconfig` command is:

```
/usr/lib/ldap/idsconfig [-v] [-i input_configfile] [-o
output_configfile]
```

In this example, the command produces verbose output and derives configuration data from the specified file:

```
# idsconfig -v -i conf-file
```

LDAP API commands are shown in Table 8.2.

Transitioning to LDAP

You can expect Sun to provide tools to assist with the transition to LDAP, regardless of which naming service you happen to be using today. The exact nature of these tools and the documents describing their use may change between the publication of this book and the time that you begin your transition to LDAP, so we encourage you to look through documents available on the www.sun.com and docs.sun.com sites.

Table 8.2 LDAP API Commands

COMMAND	FUNCTION	SYNTAX
`ldapsearch`	Search for specific entries in a directory.	`ldapsearch [options] filter [attributes]`
`ldapadd`	Add entries to a directory (user is authenticated first).	`ldapadd [options] [-f LDIF-filename]`
`ldapdelete`	Remove leaf entries from a directory.	`ldapadd [options] [-f LDIF-filename]`
`ldapmodify`	Modify entries based on changetype attribute, specified in LDIF file; you can add, modify, delete or rename.	`ldapmodify [options] [-f LDIF-filename]`
`ldapmodrdn`	Rename a directory entry.	`ldapmodrdn [options] -b "current DN" -R "new RDN" -N "new Parent"`

NIS to LDAP

For users transitioning from NIS to LDAP, there are a couple of options. One is described in a Solaris 8 vintage blueprint available online. This document, titled "NIS to LDAP Transition: Exploring," written in February 2000, describes `ypldapd`. The `ypldapd` tool, from PADL software, is a tool that emulates an NIS server (`ypserv`) using an RPC call-compatible interface. To NIS clients, `ypldapd` appears to be an NIS server. However, it stores its information in the LDAP database.

As of the writing of this book, the `ypldapd` tool is not quite ready for Solaris 9. We suggest that you check the www.padl.com site for updates on its availability.

Another option is to take advantage of the Solaris extensions for LDAP. These extensions provide similar functionality to `ypldapd`, but are implemented very differently. Instead of providing a service that accepts NIS requests from clients and translates them into the LDAP commands for use with the LDAP database, the extensions provide a synchronization service.

The extensions are provided via an add-on package to the directory service that allows LDAP to store NIS map data as well as normal LDAP entries. The plug-in communicates with the `dsservd` process that emulates an NIS server and clients communicate with `dsservd` in the same manner in which they would communicate with `ypserv`. This is depicted in Figure 8.3.

We recommend the following Sun Blueprints documents and suggest that you also be on the lookout for other documents that may have become available subsequent to the publication on this book:

- "Exploring the iPlanet Directory Server NIS Extensions" by Tom Bialaski, Aug, 2000.
- "NIS to LDAP Transition: Exploring" by Tom Bialaski, Feb 2000.
- "LDAP Setup and Configuration Guide," January 2001.

As Figure 8.4 suggests, the transition of NIS to LDAP will eventually entail a remapping of NIS information into the LDAP structure.

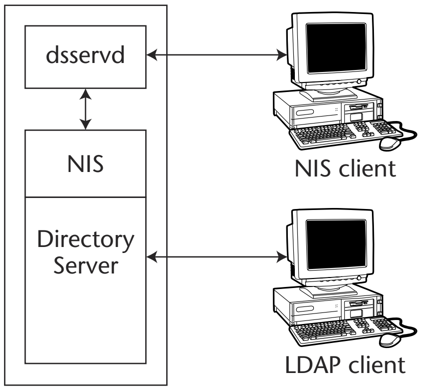


Figure 8.3 The dsservd process serving clients.

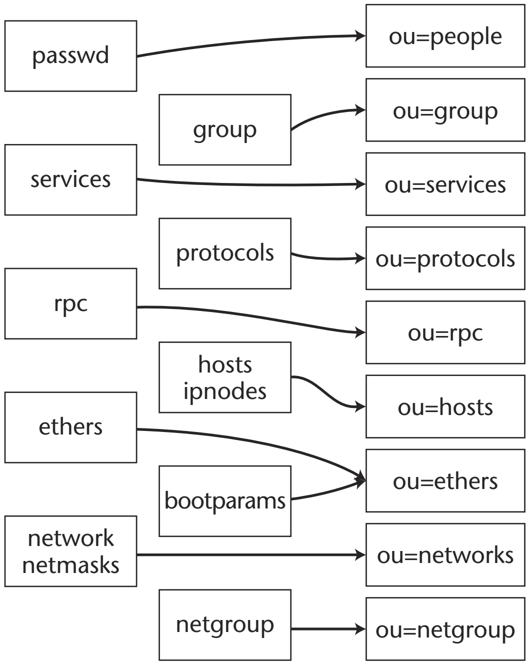


Figure 8.4 Migration of NIS maps to LDAP.

NIS+ to LDAP

A lot of attention is being focused on the NIS+ to LDAP transition. A fairly recent document entitled "NIS+ to LDAP Migration in the Solaris 9 Operating Envrionment: A Technical White Paper" is available to assist with this transition. It describes tools that are shipped with Solaris 9 to facilitate the move to LDAP. Find it at http://www .sun.com/software/whitepapers/solaris9/nisldap.pdf.

To make the transition easier, the `rpc.nisd` supplied with Solaris 9 can be configured to use LDAP as its data repository. When used in this way, the normal NIS+ database acts as a cache and prevents disruptions in communications with LDAP from having any effect on service.

A transition from NIS+ to LDAP might logically progress through installation of an LDAP server, upgrading of the NIS+ master to Solaris 9, configuring the Solaris 9 `rpc.nisd` to use the LDAP server for data, and converting NIS+ clients to use LDAP. Once the conversion of the clients is completed, the NIS+ service is no longer needed.

NIS+

Network Information Service Plus (NIS+) was designed to replace NIS, and is one of the naming services available with Solaris 9 and earlier, but is unlikely to be available in subsequent releases. The motivation for developing NIS+ was to provide a number of services that NIS lacks, such as secure authentication, hierarchical naming, and efficient updating. Even though NIS+ is dramatically different in structure from NIS and there is no direct relationship between NIS+ and NIS, it can still serve NIS clients. This makes a transition to NIS+ or support of a network with both NIS and NIS+ clients extremely practicable. However, by supporting NIS clients, you relax some of the security of NIS+.

Along with the robustness of NIS+ comes a good degree of complexity. NIS+ sports a hierarchical organization and more than 20 commands. We discuss the most useful of these in this chapter.

Like LDAP, the structure of NIS+ can reflect the structure of the organization itself. This is useful both because the organization of NIS+ will seem natural and because responsibility for each division's information can reside within that division. Another reason to consider switching is that the network overhead of NIS+ is extremely small compared with that of NIS. Because NIS+ updates are made individually, rather than through pushing entire tables or maps when only a single record is changed (as with NIS), little network bandwidth is used. The third reason is security.

NIS+Users still have their standard login IDs and passwords, as they do in NIS. However, they also have an additional password, known as the *secure RPC password,* for access to the NIS+ system. For most users, the secure RPC password will be identical to the login ID and password, but this is not always the case. The secure RPC password is used to decrypt the user's secret key and can be entered via the `keylogin` command.

Basic NIS+ Objects Explained

Like LDAP, NIS+ is *object oriented.* Thus, we introduce the components of the service by first introducing the types of objects that it manages. NIS+ objects are structural elements used to build and define the NIS+ namespace. These objects are always separated by dots when they are expressed. This point will make more sense as we provide examples of NIS+ objects. There are five basic NIS+ object types.

The first type of object is the *directory object.* Directory objects are similar to Unix file system directories. Unlike Unix directories, however, they do not contain files. They can contain one or more other objects, such as table objects, group objects, entry objects, or link objects. Like the Unix file system, directory objects form an inverted treelike structure, with the root domain (root directory) at the top and the subdomains branching downward. They are used to divide the namespace into different parts. Each domain-level directory object contains the `org_dir` and `groups_dir` directory objects for the corresponding domain (the `ctx_dir` directory is present only if you are running FNS). The `org_dir` directory objects contain table objects for that domain.

The groups_dir directory objects contain NIS+ administrative group objects. Here are some examples of directory objects:

```
foo.com.
org_dir. foo.com.
groups_dir.foo.com.
```

Another type of NIS+ object is the *table object.* Tables are similar to NIS maps. They store a variety of network information. Tables may contain zero or more *entry objects.* There are a total of 17 predefined table objects. Tables can be administered with the nistbladm and nisaddent commands. Table entry objects form a row in the table, and each row stores one record. Here are some examples of table objects:

```
passwd.org_dir.foo.com.
hosts.org_dir.foo.com.
```

Entry objects look like this:

```
[name=user1],passwd.org_dir.foo.com.
```

A fourth type of NIS+ object is the *group object.* These are NIS+ namespace administrative user groups. They permit controlled access rights to namespace modification on a group basis. They are administered with the nisgrpadm command. Here is an example of a group object:

```
admin.groups_dir.foo.com.
```

The last type of NIS+ object is the *link object.* Link objects are pointers to other objects. In other words, they are similar to symbolic links in Unix. They typically point to table or directory entries. Link objects are administered with the nisln command. For example:

```
nisln hosts.foo.com. hosts.dragonflyditch.com.
```

How NIS+ Works

The NIS+ namespace is hierarchical, similarly to that of DNS and the structure of the Unix file system. As with DNS, this structure allows the name service to mimic the configuration of the organization, and is divided into subdomains that can be administered independently. Still, hosts anywhere on the network can access information in other domains if they have the appropriate permissions.

The primary server at each domain is called the *master server* for that domain, and the backup servers are called *replicas.* The top-level server for the overall domain is called the *root master server.* It can also have replicas. Figure 8.5 illustrates the relationships between these systems.

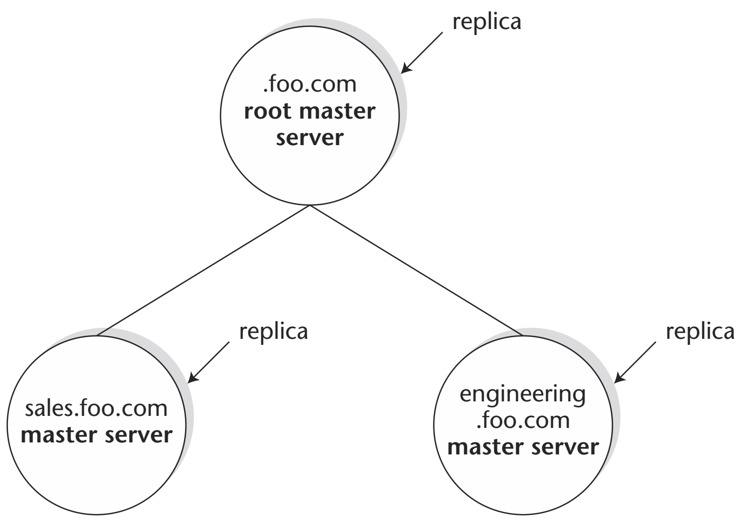


Figure 8.5 Masters and replicas.

Unlike DNS, NIS+ stores information about many aspects of a network, not just the hosts. More like NIS in this respect, this information stored in a set of 16 standard tables that constitute a special database (see Table 8.3). The master and replica servers run the server software, and both maintain copies of the tables.

Table 8.3 NIS+ Related Files

TABLE	CONTENTS
`hosts`	IP address and hostname of each system
`bootparams`	Location of the root, swap, and dump partition for diskless clients
`passwd`	Password information (username, UID, GID, home directory, etc.) for users
`cred`	Credentials for principals
`group`	Group name, group password, group ID, and group members
`netgroup`	Groups of hosts or users
`mail_aliases`	E-mail aliases
`timezone`	Time zone for hosts

(continues)

Table 8.3 NIS+ Related Files *(Continued)*

TABLE	CONTENTS
`networks`	Networks in the domain and aliases (canonical names)
`netmasks`	Netmasks associated with networks and subnetworks
`ethers`	Ethernet addresses of hosts
`services`	IP services and related ports
`protocols`	Protocols
`RPC`	RPC services
`auto_home`	Home directory map (used by automounter)
`auto_master`	Master map information, for example, list of other maps (used by `automounter`)

As with NIS, changes are made only on master servers and are then propagated to the replicas. This is true of the master server of the root domain (not to be confused with the root domain in DNS), as well as the master server of any subdomain. Storage of NIS+ data files and commands is outlined in Table 8.4.

NIS+ includes a sophisticated security system that guards not only the information it contains but also its structure. It uses this system to determine whether requests should be answered and what information any particular user is allowed to access or modify. This is in marked contrast to NIS's permissive servicing of any `ypcat` request.

Table 8.4 Location of NIS+ Components

DIRECTORY	ON	CONTAINS
`/usr/bin`	Any host	NIS+ commands
`/usr/lib/nis`	Any host	NIS+ administrative commands
`/usr/sbin`	Any host	NIS+ daemons
`/usr/lib/`	Any host	NIS+ shared libraries
`/var/nis/data`	NIS+ server	NIS+ server data files
`/var/nis`	NIS+ server	NIS+ working files
`/var/nis`	NIS+ clients	Machine-specific data files

Updates to the master server are made immediately. However, to be efficient, the master batches the changes before sending them to replicas. If no other changes occur within the next 2 minutes or so, the master stores the changes on disk and in a transaction log. The transaction log contains both the updates and time stamps. The updates are actually copies of the changed objects. Once the log has been updated, the master sends a message to each of its replicas. The replicas respond with their own timestamps—indicating the time the last update was received from the master. The master then sends all updates that have occurred since that time. This process ensures that any replica that may have been down for a while will be updated properly when it is again available. Once the replicas have been successfully updated, the master clears its transaction logs.

If a new replica is brought online, the master will notice that its time stamp is earlier than its oldest log entry and will perform a resynchronization operation. It will transfer a full set of objects and other information to the replica. Generally, both master and server will be unavailable until the process is completed. Messages indicating that the server is busy should be expected. For this reason, it's a good idea to bring replicas online during nonprime hours.

We suggest that you transition your NIS+ servers to LDAP after making careful plans for how you will manage the transition. Please read additional documents to bolster your knowledge before proceeding.

NIS

NIS is a general-purpose service for providing information across a network of systems. Unlike DNS, for example, it not only provides information on systems and IP addresses, but also contains information about users, applications, protocols, and so on. It can also be expanded to contain other information as well.

NIS is composed of a set of database files, in a particular format called `ndbm`, and a group of commands for sharing, requesting, and modifying the information stored in the database files.

From a system administrator's perspective, NIS is easy to administer because the process involves updating simple text files and then updating the shared files with a simple "make" command-`ypmake`. Years ago, NIS updates were fairly slow compared to NIS+ updates because entire maps were pushed instead of single records when updates were made. In recent releases of Solaris, however, the transfer daemon, `ypxfrd`, has become as much as 10 or 100 times faster and NIS updates are now fairly efficient.

NIS was designed as a method for sharing critical system files across the network and to maintain consistency across these systems. Files commonly stored in NIS include `passwd`, `hosts`, `automounter` files (`auto.master`, `auto.home`, etc.), services, and so on. The client packages for running NIS are `SUNWnisr` and `SUNWnisu`. The server packages are `SUNWypr` and `SUNWypu`.

There are four major differences between NIS and NIS+—the *name space* itself, *security*, the *update size,* and *key-value pairs.* While NIS+ uses a hierarchical name space, NIS uses a flat name space and stores data in *maps* (as opposed to tables). While this allows NIS+ to scale and evolve without becoming unwieldy, it restricts NIS to a fairly inflexible operational model.

Because NIS was developed in the years prior to the advent of the public Internet, security is pretty much nonexistent. All maps in NIS are transmitted unencrypted over the network. Also, the maps are easily accessible and can be decoded by anyone with rights to read the files. These things are not true of NIS+.

Incremental updates are another bonus of using NIS+. While NIS requires entire maps to be updated and transmitted, NIS+ allows incremental updating. If you have a password file with 24,000 entries, for example, NIS distributes the entire map to its slave servers even if only a single entry is changed. With NIS+, *only* the changed password entry is transmitted.

For those of you who came from a SunOS 4 (i.e., Solaris 1) background, NIS under Solaris works almost exactly the same way as it did then. However, until recently, obtaining the proper packages to set up NIS was not an easy task. Until release 2.6, Solaris did not come with an NIS server, and the unbundled product NIS Toolkit (`SUNWnskit`) needed to be installed to obtain this functionality. This toolkit can still be obtained from Sun's SunSolve Web site, although upgrading to a newer version of Solaris is highly recommended. Starting with Solaris 2.6, NIS is once again a part of the base operating system.

NIS is meant to operate in something of a hierarchical fashion, although it uses a flat name space. There is a machine that is called the *master* that holds all of the original files. There are other machines that hold replicated maps received from the master; these machines are called *slaves.* Then there is a group of machines, called *clients*, that utilize those maps. From the client's perspective, there is no difference between the master and a slave server. A client will bind to any of the servers it can (generally whichever answers first when it first requests service and binds). Also, interestingly enough, the master does not have to be a client in order to serve the maps. The slaves, however, need to be clients. The maps are all served to the clients via the `ypserv` daemon. The NIS maps are all pertinent to a single domain—not to be confused with DNS or NT domains. Let's look at each type of machine and the role that it plays in more detail.

The NIS Master

The master provides all the files that are converted into NIS maps and that, as maps, referenced by clients. The source files are not stored in any special location (though most of them are in `/etc`). Rather, the existing system files are used unless otherwise specified. In other words, you don't have to set up a pile of files in preparation for running NIS. The master will make use of a subdirectory called `/var/yp`. This directory contains the makefile and other files used to create and transfer the maps to the clients. Under this directory, a subdirectory with the domain name is created. These files are then converted into the database files—called *maps*—and stored in DBM format under the `/var/yp/{domainname}` directory.

Here is an example of setting up an NIS master:

```
# /usr/sbin/ypinit -m
```

In order for NIS to operate successfully, we have to construct a list of the NIS servers. Please continue to add the names for yp servers in order of preference, one per line. When you are done with the list, type a Ctrl+D or a return on a line by itself.

```
next host to add:  myhost       ♦ The master server
next host to add:  myhost2      ♦ The slave server
next host to add:  ^D
```

The current list of yp servers looks like this:

```
myhost
myhost2
Is this correct?  [y/n: y]  y
```

Installing the yp database will require that you answer a few questions. The questions will all be asked at the beginning of the procedure. These include whether or not you want to quit on nonfatal errors (we always answer "n" for no) and whether the prior contents of `/var/yp/mydomain` can be removed (we always answer "y" for yes).

If there are running slave `yp` servers, run `yppush` for any databases that have been changed. If there are no running slaves, run `ypinit` on those hosts that are to become slave servers.

The NIS master server has now been set up.

Changing any of the maps is as simple as editing the original file and then changing directories to the `/var/yp/domainname` and issuing a `/usr/ccs/bin/make` command. This will look for any changes in any of the source files listed in the makefile, and update the appropriate maps. After the maps are updated, they are transferred to any slave servers utilizing `ypxfr` as a transport.

The NIS Slave

The machines designated to act as slave servers must have the IP address and the name of the master NIS server in their local `/etc/hosts` files. Without this, `ypinit` will not run. If at any time the IP address or name of the master changes, all of the local `/etc/hosts` files will need to be changed, as well.

Next, the NIS domain name needs to be placed in `/etc/defaultdomain`. This file is read by several `rc` scripts when determining which daemons to run at system startup. To continue setting up the machine without rebooting, issue the `domainname {domainname}` command. Without it, the following error message will appear: `The local host's domain name hasn't been set. Please set it.` Make sure the domain name is the same as that of the master.

At this point the slave server must be bound to the master in order to enumerate the maps. This process is illustrated by the command below.

```
/usr/sbin/ypinit -c
```

In order for NIS to operate successfully, we have to construct a list of the NIS servers. Please continue to add the names for yp servers in order of preference, one per line. When you are done with the list, type Ctrl+D or a return on a line by itself.

```
next host to add: myhost
next host to add: myhost2
next host to add: ^D
```

CHANGING YOUR MASTER SERVER TO ANOTHER MACHINE

If the old master is no longer to be an NIS server, it makes things interesting. Any systems that are clients of the old master will need to have the new NIS master added to their local `/etc/hosts` files and have new bindings created. The best way to do this is to `rdist` a new copy of `/var/yp/binding/{domainname}/ypservers` to all client machines. On those machines binding to the old server, `ypbind` will also need to be restarted once the file is modified for the modification to take effect. To make these changes easily, `rdist` can be used, providing you have at least one host that has the ability to `rdist` files to all of the clients. Here's what you need to do:

First, edit the `ypservers` file so that it contains the names of the servers that will be providing NIS services—including the new master. Then, `rdist` the file to the clients only. Following is a sample `Distfile`. The command `rdist -f Distfile` would be the command used to initiate the changes.

```
HOSTS = ( myhost2 myhost3 myhost4 myhost5)
FILES = (/var/yp/binding/mydomain/ypservers)
        ${FILES} -> ${HOSTS}
        install -R;
        special /var/yp/binding/mydomain/ypservers
"/usr/lib/netsvc/yp/ypstop"  ;
        special /var/yp/binding/mydomain/ypservers
"/usr/lib/netsvc/yp/ypstart";
        special /var/yp/binding/mydomain/ypservers "/bin/echo
`172.29.10.1 myhost6 >> /etc/hosts";
```

Once there is no fear of frozen clients, and no clients still pointing at the old master, it is time to set up the new master. Make sure that all of the source files are replicated from the existing master to the new master. Then, on the new master machine, reissue the `ypinit -m` command and follow the master process just described. Once everything is set, `/usr/lib/netsvc/yp/ypstart` can be called to start the new master.

The current list of yp servers looks like this:

```
myhost
myhost2
Is this correct?  [y/n: y]
```

Now we can bind to the master with the following command:

```
#/usr/lib/netsvc/yp/ypbind &
```

At this point, the server knows the domain it is in, and is acting as a client to the master. It can now be set up as a slave. `ypinit` will not prompt for a list of other servers as it does when you create the master server, nor will it run `ypbuild` again. This is why we needed to run `ypinit -c` initially.

```
/usr/sbin/ypinit -s myhost
```

Installing the YP database will require that you answer the same questions about quitting and overwriting `/var/yp/mydomain`. Again, we always answer "n" and "y," respectively.

TIP Once all of your NIS servers are set up, it is a good idea to make sure they bind to themselves and themselves only. This way, if anything happens to one NIS server, services on all the others will still function. Many times we have seen cases in which systems bound to an NIS server do not automatically rebind to another server after a failure. For the master, it is best to *not* run `ypbind`. Its `/etc/nsswitch.conf` should be set to use files and dns only.

`ypinit` then calls the program `ypxfr`, which transfers a copy of the master's NIS map set to the slave server's `/var/yp/{domainname} directory`.

At this point, the slave server will have been successfully set up. This server will function exactly like the master, with the exception that it cannot be used to generate the NIS maps. To the clients (discussed in the next section) there is no difference between the master and a slave.

The NIS Client

Setting up NIS clients is a straightforward process. First, the domain name needs to be set and inserted into the `/etc/defaultdomain` file. Then, the IP addresses and names of the NIS servers need to be added to the local `/etc/hosts` file. Next, the `/usr/sbin/ypinit -c` command needs to be run. You will be prompted for the names of all of the NIS servers to which this client can bind. Once this is set, the /etc/nsswitch.conf needs to be modified in order to utilize NIS. The system ships with a default nsswitch.nis file that looks like this:

```
passwd:      files nis
group:       files nis
# consult /etc "files" only if nis is down.
hosts:      xfn nis [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
protocols:  nis [NOTFOUND=return] files
rpc:        nis [NOTFOUND=return] files
ethers:     nis [NOTFOUND=return] files
netmasks:   nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey:  nis [NOTFOUND=return] files
netgroup:   nis
automount:  files nis
aliases:    files nis
# for efficient getservbyname() avoid nis
services:   files nis
sendmailvars:   files
```

You will note that for password, group, automount, aliases, services, and sendmailvars, files is the default. This file can simply be copied to /etc /nsswitch.conf, and NIS will be consulted first for the remaining files.

To start the NIS client daemon, ypbind, /usr/lib/netsvc/yp/ypbind or /usr/lib/netsvc/yp/ypstart can be used.

To determine the server to which the system is currently bound, the ypwhich command can be used. This command simply echoes the name of the NIS server from which the client is enumerating maps. To change the server that the client is bound to, the ypset command can be used. In order for this command to function on Solaris machines, the initial ypbind daemon must be started with either the -ypset or -ypsetme options. Neither of these are recommended, because allowing a system to be bound to any machine is extremely insecure. The difference between the -ypsetme and -ypset options is that -ypsetme will only function on the local machine, whereas -ypset will allow anyone from any machine to change the bindings. Use these options with care, or preferably not at all.

NIS Commands

Getting information from NIS is frighteningly easy. Simple commands such as ypmatch, ypcat, and ypwhich are part of the NIS toolset.

To list all the values contained in a map, enter:

```
ypcat mapname
```

To list both the keys and the values contained in a map, enter:

```
ypcat -k mapname
```

Note that the ypservers map is one of the few maps with keys but no values.

To list all the map aliases, which are simply shortcuts to the map names, enter any of the following:

```
ypcat -x
ypwhich -x
ypmatch -x
```

They all give the same results. To list all the available maps and their master, enter:

```
ypwhich -m
netid.byname myhost
auto.home myhost
auto.master myhost
timezone.byname myhost
netmasks.byaddr myhost
... and so on
```

It is interesting to note that there can be multiple masters in a single domain. myhost could be the master for the passwd, group, and services files, while myhost2 could be the master for the auto_master and auto_home maps. In order for this to work, all of the masters need to send their maps.

To list the master server for a particular map, enter:

```
ypwhich -m mapname
```

To match a key with an entry in a map, enter:

```
ypmatch key mapname
```

Most often this command is used for finding information in the password or hosts file. ypmatch evan passwd would return the password file entry for login ID evan. Some of us old-timers prefer combining ypcat with grep. It seems more like raw Unix. ypcat passwd | grep ^evan will perform the same function.

TIP If you keep phone numbers in the gecos field password file, ypcat | grep is a quick way to do a phone lookup!

Hardening NIS

NIS is not a particularly secure distributed name service. Unaltered, it will provide the information from its tables to any system aware of its domain name. There are two things that a system administrator can do to make NIS more secure. One is to list trusted hosts and network addresses in the /var/yp/securenets file. Another is to use secure RPC. In any case, it is wise to omit root and other system accounts from the tables that are shared via NIS.

The securenets file should reside on the NIS server, in the directory with the NIS database files, /var/yp. It defines the hosts and networks that will be granted access to information about the domain. This file is read by ypserv and ypxfrd.

The securenets file contains a series of netmask and network addresses such as these:

```
255.255.0.0         10.4.0.0
255.255.255.0       10.3.2.0
```

Secure RPC will ensure that users are authenticated whenever a process they run makes a remote procedure call. Secure RPC can use Unix, DES, or Kerberos authentication mechanisms.

Summary

Regardless of what name service you are using today, the direction that you need to head is LDAP. If you are still using NIS and are not ready to transition to LDAP, consider hardening it as we've explained in the preceding section. If you're using NIS+, plan to migrate to LDAP in the next year or two; NIS+ is not likely to be available in any release of Solaris after Solaris 9.

If you can, set up an LDAP server and a client or two in a test environment. Run through your transition plan in the lab environment long before you attempt to migrate your production systems, giving yourself plenty of time to get used to the LDAP services and the configuration of both servers and clients.

In this chapter, we introduced LDAP as a Solaris naming service, compared it with other naming services available on Solaris, and provided overviews of LDAP, NIS+, and NIS.

PART

II

Managing Your Systems

Though the set of available tools for managing networks has changed considerably since the first edition of this book, the challenges remain the same. The complexity of our networks, systems, and software is ever increasing, and our limited ability to keep pace means that system administrators are forced to remain in a steady state of learning how to configure and manage technology.

To successfully manage a network of systems, processes, and users, we need to find ways to take the day-to-day ho-hum challenges out of our daily routines so that we can concentrate on the unusual. This involves automating as much as is reasonably possible. Creating scripts as a way to avoid repetitive work is the single most important task of systems administration. On the other hand, creating scripts isn't the whole job. We also have to remember what each script does, organize our scripts in some manner that allows us to easily use and update them, and, when appropriate, run our scripts through services like `cron` so that we don't even have to invoke them to reap the benefit.

We also need to manage system security and take advantage of technology that provides high availability.

Keeping our systems well organized, well monitored, and available is the key to doing a good job and having a good job.

In this part of the book, we discuss many aspects of keeping your systems well organized and usable. In Chapter 9, we address system and network monitoring. We provide some information on commercial network monitoring software as well as monitoring that you can do with scripts and some well-chosen tools. In Chapter 10, we examine performance tools and the resources that they monitor. In Chapter 11, we look at volume management—a technology that has radically changed the way file systems and disks are managed. In Chapter 12, we offer some advice on how and what to automate to make your systems more manageable. Chapter 13 provides an introduction to system security—including a lot of recommendations to help you nail down your systems quickly. Chapter 14 introduces high availability and tells you what you can do, even on a small budget, to keep your network services up and your systems available most of the time.

CHAPTER

9

Monitoring Your Environment

Proactive monitoring means looking for trouble before it finds you. With reference to computer environments, monitoring is almost always proactive. Watching for evidence of emerging or potential problems, checking available disk space and the health of essential processes, and probing systems to ensure that they are reachable are all examples of detecting problems before they disrupt the workflow on your network.

Monitoring is not something you should cavalierly put off until there's nothing else on your full plate of things to be done. As soon as your computer environment is up and running, it is time to start thinking about protecting it against all the things that can go wrong. A good place to begin is to ask yourself a series of questions. What kinds of problems are possible? Which of these problems will cause serious disruption to your company's productivity? What information do you need to know about the status of your systems to keep them running smoothly? When do you need to know it? How do you want to be notified? These are questions that must be answered fully before you can put any comprehensive monitoring plan into place.

This chapter suggests some answers to these questions, discusses the services that network management software offers, and provides you with some useful examples of monitoring and notification scripts.

Why Monitor?

Monitoring is essential because every network, with very few exceptions, is far too complicated for anyone to manage without some kind of automated assistance. In fact, automation is implied in the term *proactive monitoring*. Anyone monitoring a system or a network manually (i.e., without the help of scripts and other software tools) is being reactive, not proactive. Any break in normal processing, any kind of emergency or failure, is likely to steal his or her attention completely from all the other systems and services that need to be watched.

The alternative to monitoring is waiting for things to break and then responding as needed to the resultant emergencies. Most of us, in spite of our best intentions, will find ourselves in this mode from time to time. For the better prepared of us, however, these times are rare. When they occur, we usually have the luxury of being able to focus our attention on the problem at hand, confident that other aspects of our operations are not running on the very edge of usability and are likely themselves to fail.

The key elements of monitoring are as follows:

- Reducing the amount and complexity of data that you need to review. Examples include log files, error messages, quota reports, and the output from scheduled `cron` jobs.
- Attaching virtual probes to essential services so that you are notified immediately when they become unavailable. Examples of services you might want to monitor in this way include Web servers, database servers, and mail servers.
- Maintaining big-picture connectivity and performance statistics, often through the use of graphical displays of the network being monitored. No service is useful if you cannot access the system on which it is running.
- Assisting in the detection and isolation of hardware and software faults.

Monitoring can take the form of very simple scripts that are invoked through `cron`, or it can involve large commercial packages and dedicated high-powered workstations. No matter how much computer power or software you throw at the problem of network and system monitoring, however, some manual intervention is required. The analogy of a tree in the forest that falls without anyone hearing is a good one. If no one reads the script-generated reports, looks at the monitoring station's graphical display, or listens for the alarms, is the network being monitored? We think not. For us, therefore, the term *proactive monitoring* implies several factors—automation, detection of problems before they become critical, and reception of reports and alarms by someone who understands their significance.

How much to monitor, how frequently, and at what expense depends on the size and complexity of your network, the degree to which the services it supplies can be considered critical, the window within which your services must be available, and the size and skill of your support staff.

It's easy enough, perhaps, to check disk space using a `df -k` command, but what if you're responsible for more than 60 servers, all of which have several gigabytes of highly volatile disk space, or thousands of clients? Can you effectively keep track of that much disk space spread across that many systems? How often do you need to check if file systems are dangerously full? How do you do it? Do you Telnet to each system and run the `df` command? Do you set up a `cron` job to send e-mail to you any time any of the disks on any of the system is more than 90 percent full? Do you set off a script to truncate or compress log files?

Even if you have an answer for managing disk space, what about all the other things that can go wrong? Do you scan messages files for warning messages from your system telling you that there have been memory errors or disk errors? Do you scan `sulog` for signs that someone knows someone else's password, knows the superuser password, or is trying to guess passwords? Do you routinely look at network traffic? Do you have any idea how high collision rates and error rates are on your clients' network interfaces? Are your users over their disk quotas? Do you know when mail is backing up in your spool directory? Do you check root's mail file now and then, or is mail addressed to root delivered to you? Have accounts for people who have left your company been properly disabled? Are your backups being completed properly? Do your systems have the proper patches installed? Are your users' environments set up properly?

So many factors contribute to the overall health and well-being of a network that it is hard to name them all, never mind watch them all. Determining which of the large number of things that can fail, and should be monitored, is a task in itself. If you elect to watch everything you can think of without the benefit of a graphical network display, you can easily be barraged by e-mail or paged so often that you have trouble telling the minor emergencies from the major emergencies.

Commercial network management packages such as HP Openview, Sun Net Manager, Netview 6000, and Robomon are all wonderful products, but not everyone can spend thousands of dollars on a framework for system monitoring. Small companies or offices are not likely to be able to afford the tools or the personnel to support a fully functional monitoring station. We will talk about commercial network management and monitoring systems in enough detail to describe their underlying protocols, basic services, configuration, and ongoing maintenance. But don't lose heart if your budget is small or your boss does not understand how difficult it is to keep track of the health of a network of busy systems. There are many things you can do to monitor your systems directly with customized scripts, Internet access, email, and a pager. We will provide some recommendations for publicly available tools and offer some of our favorite tips and scripts for constructing the poor person's network monitor.

Monitoring Tools: Build, Buy, or Get for Free?

In every environment, there is a multitude of items that should be monitored on a continuing basis. For the base operating system, such things as disk space, swap space, CPU usage, and warning/error messages that are written into the `/var/adm/messages` file are a few that come to mind as needing to be checked periodically. For applications such as DNS, directory services, and Web services, you may have to ensure that these services are operating around the clock.

The build or buy decision is not as mutually exclusive as it might appear. Almost anyone desiring a home-brew network management system today will start with shareware. The emergence of Sun Management Center (formerly SyMON) as a free tool with platform-specific add-ons (not free) presents a third solution that may be hard to resist, especially if portions of this technology are already working their way into the core operating system.

One thing to remember is that, in every case, there is a chance to spend a boatload of money on a product that will perform the needed monitoring for you. However, there are two things to keep in mind. First, implementing your own monitoring system may not only prove a rewarding experience, but will also allow you to customize your monitoring and alerting process as no commercial program will. Second, even the most fully featured network monitoring software will require some setup and customization as the scope of your monitoring responsibilities changes with changes in your network.

Another thing that you should keep in mind as you make decisions about whether to buy or build your monitoring tools is that it is hard for many companies to get over the "freeware is bad" syndrome. Utilities such as wget, Lynx, and Perl are the cornerstones for most script-based monitoring. If your management mistakenly believes that basic tools are unreliable and that, conversely, anything that arrives in shrinkwrap is reliable, you might have a hard time getting the backing to "roll your own" or recognition for your efforts afterward.

Some of the major differences between commercial and home-brew monitoring systems involve the approach taken with respect to both protocols and delivery mechanisms. Commercial packages are generally built on top of specially crafted protocols designed to request and obtain information about the systems and processes you are monitoring without overwhelming the network in the process. They generally deliver their reports to graphical displays representing the collection of systems being monitored. Most home-brew systems, on the other hand, rely on a collection of scripts and use a combination of pagers and email to deliver their findings to the interested parties.

As was suggested earlier, one of the most important factors to consider when making the build-or-buy decision is the size and complexity of the network you are monitoring. Large organizations might do well to have a monitoring station that is staffed 24 X 7. One of the authors once worked as an authentication engineer for a company running a global virtual private network (VPN). For this company, round-the-clock monitoring of systems spread across the globe, as well as the network connecting them, was both cost-effective and necessary. At other times in our careers, this approach would not have been justified.

Evaluating Network Management Systems

The more networks evolve into complex distributed mission-critical networks, the more there is a corresponding need for centralized administration. With the recent exponential growth in the Internet and distributed computing, along with a steady move toward heterogeneity, it is almost impossible to run a network without some level of centralized administration and monitoring.

Commercial network monitoring systems are built to run on top of protocols designed to efficiently gather data from computers and other networking devices (e.g., routers and switches) and deliver it to the management station where, presumably, some human being is watching the console for signs of trouble.

These protocols make it possible for a variety of disparate hardware to essentially speak the same language. By agreeing on a simple set of primitives for requesting and sending network information and a simple record structure in which to pack and deliver it, the systems involved provide the basis for the exchange and for the displaying of network information at the monitoring station.

Basic Terminology

Before we go any further into outlining the various protocols and touching upon some of the more prominent network management systems in use today, let's describe the basic operations of network management systems and establish a vocabulary with which we can describe them. The terms described in this section are used primarily to describe network management systems that are based on a protocol called Simple Network Management Protocol (SNMP). Solaris 9 also provides support for another network management protocol—the Web-Based Enterprise Management (WBEM) protocol. We describe what this support entails and how the two protocols relate later in this chapter.

Agent. A network management agent is a process that runs on a client system or network device. An agent's role in life is to listen for requests from the network management station, retrieve the requested information, and then reply. Agents have access to information about the operational status and configuration of the device and are able to respond to requests. They can also control the devices by managing the configuration information. Agents across different devices respond to the same set of commands with responses of the same general form regardless of the type of device or the complexity of the device's internals.

MIB. MIB stands for management information base, though you will always hear it pronounced as a single syllable word rhyming with "bib." A MIB contains current and historical information about the local configuration and network traffic. Each agent maintains a base of information about the local system, while the management station maintains a global MIB with a summary of information from all of the agents.

MIBs are extremely important, more so than they might initially appear to be. In fact, they determine the type of information that is maintained by the agent and relayed when a management station requests it with some type of `get` command. The most common MIB in use today is MIB-II, but there are many additional MIBs, some of which are proprietary. The differences are needed to accommodate the additional functionality of the devices being managed and are not a matter of concern.

Proxy. A *proxy* is an agent that provides support to older devices that do not understand current protocols or any network management protocol at all. In other words, a proxy exists outside of the managed device and provides a means for managing legacy systems.

Alarm. An *alarm* is a general-purpose term used to describe the report of a network event. Alarms are also called traps.

Polling. *Polling* is a periodic action performed by a network manager to request information from agents.

Discovery. *Discovery* is the process by which a management station automatically determines the systems and devices on the network needing to be monitored. In general, the manager sends out requests for descriptive information from each of the devices and then uses this information to assign icons and characteristics for the graphical display.

Network manager. The *network manager*, or *network monitoring station*, is the system on which network status information is collected and displayed. This generally involves a fairly powerful graphical representation of the network using icons for each of the network entities, or collections thereof, being monitored. It can, on the other hand, be a simple command-line system on which the primitives are typed.

Network Management Protocols

The most widely used and simplest of network management *protocols* is the Simple Network Management Protocol (SNMP). SNMP was intended, initially, as a stopgap measure—a tool to provide some standardization to network management until an expected transition to an OSI-based network framework had been accomplished. While more sophisticated protocols were being specified and the move to OSI was first stalled and later abandoned, SNMP was further developed to accommodate pressing needs. A number of versions now exist and yet, in spite of the existence of other protocols, SNMP continues to dominate the network management scene.

The basic operation of SNMP, and any network management system in fact, involves the agent processes, which collect information about the system on which they are running and provide it to management systems, either in response to explicit requests (polls) or in response to the severity of things to be reported. Agents can also respond to requests from the management system to alter parameters on the client. Management systems are generally equipped with a graphical display of the network being monitored.

SNMP, therefore, defines a protocol for the exchange of information that relates to the operational status and configuration of network devices, the format for representing this information, and a framework involving agents and managing hosts. Current SNMP versions include SNMPv1, SNMPv2, SNMPv2c, and SNMPv2 usec, and SNMPv3, which provides security enhancements.

The four basic operations, or primitives, that SNMP uses are:

get. Used by a network manager, requests information from an agent's MIB.

set. Used by a manager, sets a parameter value in an agent's MIB.

trap. Used by an agent, sends an alert to a manager.

inform. Used by a manager, sends an alert to another manager.

If you delve further into the protocol, you will see these basic operations reflected in actual commands that are run—for example, the `get` operation is expressed in the `Get`, `GetNext` and `GetBulk` primitives.

Competing and complimentary protocols include CMIP, RMON, and WBEM. RMON is the remote monitoring standard. The goal of RMON is to allow the use of networking monitoring devices that can measure certain aspects of a network without interfering with it. An RMON probe is a network device that runs an RMON agent using the SNMP remote-monitoring MIB. It continuously monitors traffic and tracks statistics such as the number of packets, octets, broadcasts, and percentage of network utilization.

At present, there are a variety of technologies in common use in network management applications:

- SNMP and CMIP management protocols for network management
- RPC (remote procedure calls) for systems management
- OMG CORBA for emerging systems and network management frameworks

Network Management Products

Although we don't want to endorse any particular product (we're too busy to try them all!), we feel it's appropriate to mention some of the commercial network management products that might answer your needs, and explain some of the features that may be valuable to you.

Solstice Domain Manager and Solstice Site Manager (both fall under the SunNet Manager umbrella) provide centralized management for large multisite or small networks. SunNet Manager was an early entrant into network management and has grown in features and sophistication. Current versions feature cooperative consoles (for peer-to-peer cooperative management consoles) and support for heterogeneous networks (including accessing Novell NetWare Management Agent). You can choose to fully centralize the management load or spread it around to different groups. Using these tools, you can manage a wide range of system resources, including: communication protocols, network devices (switches, hubs, bridges, routers, etc.), applications, network services, and operating system resources.

Other tools to investigate include Sun Management Center (which we discuss in more depth later in this chapter), HP OpenViewNet/Metrix, Tivoli, CA Unicenter TNG, Enlighten, Halcyon, Robomon, and Heroix Management Suite. Since these products will all provide many of the same features with different strengths, weaknesses, and price tags, only an evaluation with respect to your network, needs, and budget will yield the proper choice.

Any network management system should provide you with a basic set of functions. In our opinion, this set includes the following:

Discovery. System administrators are far too busy to manually build up the information base required for network management.

Problem detection. The system should detect and report device and network failures.

Problem resolution. The system should have a means of initiating scripts or taking some action to repair problems (when possible).

Notification. The system should be able to notify you in a number of ways, depending on the severity of a problem. These should include visual cues on the manager display as well as email and paging.

Reporting. The system should be able to prepare a report on the general health of the network, problems detected, and actions taken within a given time frame.

Display. The network manager console should be able to assume different views of the managed network and focus in on segments or individual systems when problems occur.

Scalability. Whatever system you select, it should work on both the small and large scale or work seamlessly with other tools to provide you with integrated management of your network.

Additional functions that we think add significantly to the value of network management include trend analysis and correlation of events, to support sophisticated problem analysis. Automated trouble ticketing for problems requiring intervention by the system administrator or support staff is especially useful in large networks.

Poor Person's Network Monitoring

For those of us without the resources to purchase network-monitoring software, there are many tools and techniques available to help us perform adequate monitoring, albeit in a less comprehensive and integrated fashion.

Deciding what needs to be monitored, how this can be accomplished, and how we want to be notified when something needs attention are the first steps.

Deciding What You Want to Monitor

There are a number of things that you will undoubtedly want to monitor. They fall into several broad categories:

Critical processes. You want critical processes to stay up and responsive. These include most system daemons (network daemons, naming services, file systems, printers, Web servers, logging services, and critical applications).

System availability. You want system resources to be available and usable. For example, disks need to both be available and have available space. Many monitoring systems issue warnings once a certain threshold (say 90 percent) has been reached or exceeded. Better monitors will also issue a warning if space is filling up quickly (especially if it's happening more quickly than usual).

Connectivity can be as critical as whether a system is available. If a system and its processes are running, but users cannot access them, they are of little value. Most system management systems monitor connectivity across the network. If a certain system or a network segment becomes unreachable, you can expect a visual alarm and maybe more. If you have systems that must be reachable, monitoring them from another system (or two) is a good idea.

TIP Don't forget to consider the possibility that a system used to monitor other systems may itself go down. Don't forget to consider that a process set to warn you about problems might itself be in trouble.

Signs of trouble. You want to be on the watch for certain occurrences, such as signs of a break-in and new or impending hardware failures.

Performance monitoring will help you detect problems of many types—from runaway processes to network attacks meant to disable your services. It will also help you to make decisions to move services or upgrade systems in order to provide an acceptable level of service. See the *Performance Monitoring* section later in this chapter for more information.

Log files can provide extensive information on how your systems are being used, but only if you look at them. Our preference is to extract certain message types from our log files, digest them into statistics, and mail ourselves reports. Log files grow too quickly and are far too numerous for any of us to do a proper job of scanning and understanding all that they have to tell us. Taking the alternate route, ignoring them completely, is worse than trying to read them from top to bottom. Fortunately, there are some fine tools to assist you in monitoring log files. We describe some of these in the *Log Monitoring* section later in this chapter.

Deciding How You Want to Monitor

If you decide to use a commercial network application, you'll have a head start on deciding what you want to monitor. Every tool that we have used or looked at provides a number of built-in templates for such things as disk space, CPU utilization, critical processes, and connectivity. If you're going with a home brew approach, you'll want to come up with your own set and then determine how you can best monitor each.

For "live" monitoring (i.e., you're sitting at the screen), `perfmeter` can be useful. It does, however, require that remotely monitored systems be running the `rstatd` daemon (`rpc.rstatd`). If you watch a critical server with `perfmeter` (see Figure 9.1), you will notice right way if CPU usage hits the ceiling or the load starts to climb. Keep in mind that the number displayed along with each graph represents the *scale* of the individual panel—not the high value. A flat-line display on a panel showing a 64 is still zero.

`perfmeter` displays any or all of a number of values, as shown in Table 9.1.

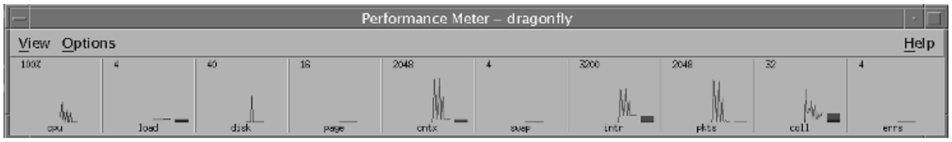


Figure 9.1 Performance meter (perfmeter).

Table 9.1 perfmeter Values

VALUE	DESCRIPTION
`cpu`	The percentage of the CPU being utilized
`pkts`	Packets per second coming/going on the network interface
`page`	Paging activity (see Chapter 1)
`swap`	Jobs swapped per second (see Chapter 1)
`intr`	Device interrupts per second
`disk`	Disk transfers per second
`cntxt`	Context switches per second
`load`	Average number of runable processes in the preceding minute
`colls`	Collisions per second
`errs`	Errors per second (received packets only)

For times when you might be there and might not, scripts are the ticket. You can write scripts that check almost any measure of system health you're interested in, and send yourself mail or even page yourself with the results.

You can also install and configure tools that monitor system resources or digest and analyze log files for you. We mention some of our favorites in the *Performance Monitoring* section a bit later in this chapter.

Deciding How You Want to Be Notified

The first thing that you need to set up is the notification mechanism. Let's assume that, depending on the severity of the alert, you will either want to be notified immediately via pager or via email (for less critical problems). We will start by looking at a script that has the ability to do both. The pager script shown below shows how easily pages and email can be sent. This script requires Internet connectivity as well as having Lynx (a text based browser) loaded on the system. The other requirement is a Web site or other service site from which paging can occur. In this example, pagenet.com is used. The pager script accepts the message using *standard in*, with the alphanumeric paging pin number as a parameter, and uses Lynx, a text based browser, to post the message to the paging system's Web site. Here is an example of how you might call this script into action:

```
cat /etc/motd | /usr/local/bin/pager 1234567
```

This command would send a page containing the contents of the message of the day file to a pagenet pager with the specified phone number. These pages are predicated upon HTML posts performed by these sites and may need to be modified accordingly to get them to function with your paging provider Web site. The Pager Script for Pagenet follows:

```
#!/usr/bin/ksh
#
phonenum="987-123-4567"
pin="1234"
echo "message>\c"
read msg
echo "Message=$msg" >> /tmp/pager.log.$LOGNAME
date >> /tmp/pager.log.$LOGNAME
echo $msg | encodedmsg=`/usr/xpg4/bin/awk '
{ for (x=1; x <= NF; ++x) if (x==1 && NR==1) s=$1; else s=s"+"$x }
END { print s }'`
for pin in $pin
do
  echo "Pin=$pin" >>/tmp/pager.log.$LOGNAME
  opts="-post_data -dump -nolist -error_file=/tmp/pager.log.$LOGNAME"
url="http://www.foopage.net/foopage/page_gen"
/usr/local/bin/lynx $opts $url >> /tmp/pager.log.$LOGNAME 2>&1 <<__EOD__
SysID=$phonenum&To=$pin&Message=$encodedmsg
__EOD__
done
echo "----------------------------------\n" >>/tmp/pager.log.$LOGNAME
```

The script can be modified for another site by replacing the URL, phone number, and pin.

Email notification is much simpler, since the `mail` and `mailx` commands already reside on your Solaris systems. If you use a command like this:

```
cat /etc/motd | mailx -s "Message of the day" emarks@foo.com
```

an e-mail will be sent to `emarks@foo.com` containing the same information that we just sent with the paging scripts.

Once your page script and email setup are complete, you can start preparing scripts to monitor your systems and page or email you when problems are detected.

Performance Monitoring

Performance monitoring is such an important aspect of network monitoring that we have added a new chapter in this edition (Chapter 10) to provide a more detailed look at the many aspects of performance. At the same time, performance monitoring is folded into many network management systems, whether home brew or commercial. In this section, we cover some of the commands and tools that you might use in building your own management station.

One of the first places to look when a system is slow is, of course, the output of the `ps` command. The more useful things to notice are:

The number of processes. Is it larger than usual? Has some system daemon spawned many copies of itself? Can you determine why this might be the case?

Unusual processes. Are there unfamiliar processes running? Are they using a lot of CPU time?

Who is running what? Are any users (including root) running processes that seem excessive?

Are some processes running at high priority? Are some processes getting too much access to the CPU to the detriment of others?

At one of our sites, a Solaris client was crawling because a user had nearly a dozen `textedit` sessions running (apparently, he didn't know to exit and was, instead, putting them in the background). Problems like these can go unnoticed until a system gets so slow that the user becomes irritated enough to complain.

There is a lot more information available about processes than the `ps` command, in its popular forms (e.g., `ps -ef`), lets on. Should you have any doubt, a quick scan of the structures used to maintain process information (i.e., `prpsinfo` and `prstatus`) will convince you otherwise. Some of this information can be extracted by variations of `ps` commands and some is better highlighted by other tools.

Once you've derived what you can from the output of `ps`, you might use the `top` command, available from various FTP sites, to gather additional information. `top`, as its name implies, reports the process most heavily using the CPU, along with additional information, including statistics on memory use and the system load (three intervals). The system load reports on the number of processes that are ready to run but awaiting their turn. If a system load is small (less than 2), your system is not likely being slowed down by processes hogging the processor. If it's approaching or exceeding 4, this is a concern. `top` also displays the nice values.

`perfmeter` is a built-in graphical tool that will give you a lot of information on a system's performance. Using `perfmeter`, you can determine whether a system is swapping or excessively context switching, or whether the network is congested or disk activity is high. If you don't use `perfmeter` often enough to have developed a gut feel for what is normal for a system, place two `perfmeters` side by side—one for the troubled system and one for a similar system performing well—and look into the obvious differences.

TIP **Comparing output on a troubled system with the output from a similarly configured behaving system will almost always provide you with some insights. Be careful in selecting the systems to compare so that your comparisons are not distorted by dramatic differences in the systems' capabilities.**

Once you've determined that a particular process is slowing down a system, you might want to find out why. There are tools for gathering relevant data for this kind of analysis. The `proc` commands (discussed in Chapter 1) provide a wealth of information about running processes, but will require an investment of time before you can use them wisely.

The truss command will produce a trace of a process, allowing you to easily see what files it opens (or tries to), what system calls it makes, and so on. In the following sample output, the particular process (lynx) tries to open a couple of files that do not exist (i.e., NOENT).

```
open ("/user/local/lib/mosaic/mime.types", O_RDONLY) Err#2 ENOENT
open (".mime.types", O_RDONLY)                       Err#2 ENOENT
open ("/home/nici/.mime.types", O_RDONLY)            Err#2 ENOENT
```

sar

The sar (System Activity Report) tool is an excellent performance package that comes with Solaris. It can be used for data gathering and is extremely helpful when one of your users complains to you that "things started getting slow yesterday around 3 P.M." Using sar reports, you can determine the cause of many performance slowdowns. You might notice, for example, that the CPU was very busy at the time, or that there was a large amount of paging activity on the system. Because sar collects a variety of system activity data and saves it for later scrutiny, it provides a good source of unbiased evidence for your analysis. See Chapter 10 for more information on sar.

Log Monitoring

While sar is great at data gathering and reporting system information, it doesn't tell you everything you need to know to determine the health of a system. You might also want to review system logs on a daily basis. The following script, if set to run at 11:59 P.M., will e-mail the day's records from the messages file from the system on which it is run. Of course, you would *not* want to do this with syslog because of its sheer size!

```
#!/bin/sh
cd /var/adm
MACHINE=`hostname`
if [ `date '+%d'` -lt 10 ]; then
   DAY=`date '+%b %d'|tr '0' ' '`
else
   DAY=`date '+%b %d'`
fi
echo "Subject: $MACHINE Log" > logmon.out
files=`find . -name 'messages*' -mtime -1 -print | sort -r`
for file in $files
do
cat $file | grep "$DAY" >> logmon.out     <- If only concerned about
done                                         certain messages, pipe the
                                             output into another grep.
/usr/lib/sendmail -f"Logger" e-mail-id <logmon.out
done
```

Another script that we recommend running at 59 minutes after every hour is called watchlog. It looks for warning messages from the past hour in the messages file, writes any it finds to a log file, and then pages the listed pagers. Warning messages are more severe than the messages commonly written to this file and, sometimes, need to be acted on immediately. By seeing these messages in near real time, you may be able to salvage data from a failing disk before it becomes unusable.

```
#!/bin/ksh -xvf
#
PAGEERM="1234567"
PAGESHS="1234566"
PAGELIST=" $PAGEERM $PAGESHS"
MYNAME=`hostname`
log=""
#
if [ `date '+%d'` -lt 10 ]; then
   DAY=`date '+%b %d'|tr '0' ' ' `
   LASTHOUR="$DAY `date '+%H'`"
else
   LASTHOUR=`date '+%b %d %H'`
fi
#
log=`grep "$LASTHOUR" /var/adm/messages|grep -i warning`
echo $log
if [ ."$log" != "." ]; then
   echo "$MYNAME LOG Alert `date +%H":"%M` $log" |\/usr/local/binpager
        $PAGELIST
   echo "Message Log Alert Logged" | logger -p local0.alert -t watchlog
-i
fi
```

In general, Solaris log files cannot be manually monitored. There are simply too many log files and too much information. Digesting the information in the log files of a single busy system could be a full-time job. Our strategies at data reduction involve focussing on the more important messages (i.e., warning messages), summarizing the data in log files by running the logs through analysis scripts (Perl is a great tool for this kind of job), and intercepting messages as they arrive using tools like `swatch` described in the next section.

swatch

One of the most useful tools that we have encountered for making effective use of system logging and system log files is `swatch`. It is extremely valuable if you manage a large number of systems—especially a large number of servers—because its log file monitoring (real time) and auditing (later analysis) are invaluable.

swatch was originally designed to monitor messages as they are being written to the log files. It works with any log files written by the syslog facility. It can also be run in an auditing mode in which an entire log is analyzed for occurrences of certain types of messages.

swatch is Perl-based and this shows. It works on patterns describing the types of messages that it is looking for while doing its analysis. Its basic processing involves patterns and actions to take when messages matching those patterns are detected. You might, for example, want to be notified immediately whenever a "file system full" message appears in your messages file.

The actions that swatch can take when a pattern is detected include:

echo. Display the message.

bell. Display the message and signal with an audible alert.

mail. Send email to the user running the program.

exec. Run a script or command.

pipe. Pipe a message to a program.

write. Use the system write facility to send a message.

ignore. Ignore the message.

The configuration file that swatch uses, .swatchrc, contains these patterns and actions. For example, a line like this:

```
/.*file system full/ mail=sysadmin,exec "pager xxxx"
```

will send email to the alias sysadmin and run our pager script if a "file system full" message is written to our log. It follows the general form /pattern/ action. You can include multiple patterns, like panic|halt, and multiple actions—as shown in the previous example.

You can also include two additional fields in .swatchrc entries. These help you to avoid being inundated with messages regarding a single problem. The third field specifies a time interval during which you do not want to take action on identical messages. The fourth further specifies the location of the timestamp in the record. Both are optional.

Swatch can be invoked without arguments or with any of the options shown in Table 9.2.

swatch was written by Todd Atkins at Stanford and can be downloaded from the following FTP site: ftp://ftp.stanford.edu/general/security-tools/swatch/.

When first configuring swatch, you should make a decision regarding your basic approach to screening messages with this tool. The two approaches are to specify the messages you want to see or suppress those you don't (using ignore). Consult the sample configuration files included with the tool.

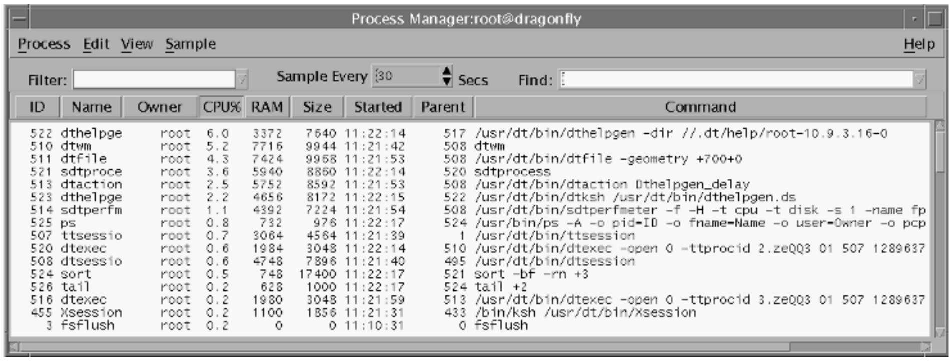
Table 9.2 swatch Options

OPTION	DESCRIPTION
t *filename*	Monitor the log file specified.
f *filename*	Audit the specified file.
p *program*	Monitor output from the specified program.
c *filename*	Use an alternate configuration file.
r *time*	Restart at the time specified.
P *separator*	Use an alternate character (i.e., instead of |) for separating patterns in the configuration file.
A	Use an alternate character (i.e., instead of a comma) for separating actions in the configuration file.
I	Use an alternate character (i.e., instead of an newline) as a record separator in the input.

Process Manager

Another useful tool for examining processes is Process Manager. Built into the default Solaris desktop, Process Manager allows you to look at running processes and reorganize them according to various criteria (e.g., amount of CPU utilization or process size). Process Manager is found in the Tools menu and is shown in Figure 9.2. By default, the processes are ordered by CPU utilization.

Process Manager also allows you to look at other aspects of a process. For example, you can display the parentage (ancestry) of a process or trace system calls.



```
ID   Name      Owner CPU% RAM   Size  Started   Parent Command
522 dthelpge    root  6.0  3372  7640 11:22:14   517 /usr/dt/bin/dthelpgen -dir //.dt/help/root-10.9.3.16-0
510 dtwm        root  5.2  7716  9944 11:21:42   508 dtwm
511 dtfile      root  4.3  7424  9958 11:21:53   508 /usr/dt/bin/dtfile -geometry +700+0
521 sdtproce    root  3.6  5940  8860 11:22:14   520 sdtprocess
513 dtaction    root  2.5  5752  8592 11:21:53   508 /usr/dt/bin/dtaction Dthelpgen_delay
523 dthelpge    root  2.2  4656  8172 11:22:15   522 /usr/dt/bin/dtksh /usr/dt/bin/dthelpgen.ds
514 sdtperfm    root  1.1  4392  7224 11:21:54   508 /usr/dt/bin/sdtperfmeter -f -H -t cpu -t disk -s 1 -name fp
525 ps          root  0.8   732   976 11:22:17   524 /usr/bin/ps -A -o pid=ID -o fname=Name -o user=Owner -o pcp
507 ttsessio    root  0.7  3064  4564 11:21:39     1 /usr/dt/bin/ttsession
520 dtexec      root  0.6  1984  3048 11:22:14   510 /usr/dt/bin/dtexec -open 0 -ttprocid 2.zeQQ3 01 507 1289637
508 dtsessio    root  0.6  4748  7896 11:21:40   495 /usr/dt/bin/dtsession
524 sort        root  0.5   748 17400 11:22:17   521 sort -bf -rn +3
526 tail        root  0.2   628  1000 11:22:17   524 tail +2
516 dtexec      root  0.2  1980  3048 11:21:59   513 /usr/dt/bin/dtexec -open 0 -ttprocid 3.zeQQ3 01 507 1289637
455 Xsession    root  0.2  1100  1856 11:21:31   433 /bin/ksh /usr/dt/bin/Xsession
  3 fsflush     root  0.2     0     0 11:10:31     0 fsflush
```

Figure 9.2 Process Manager.

The SE Toolkit

Another extraordinarily useful, though unendorsed tool from Sun, the SE Toolkit can be thought of as a performance engineer looking over your shoulder. In fact, this tool was created by two such engineers (Richard Pettit from Foglight Software and Adrian Cockcroft from Sun) and provides insights on various aspects of your system without you having to do much of anything at all. Of course, if you want to, you can delve deeply into the tool. It is, after all, a toolkit. That means you can use it as a basis for creating other tools for analyzing your systems' performance—and, indeed, a number of people have.

The tool, as the output below suggests, monitors many aspects of your system and makes recommendations. Overloaded disks, memory shortages, and network load will not escape your notice and the advice offered may help you decide how to best upgrade your systems or your network to achieve better performance.

```
/opt/RICHPse/examples/mon_cm.se Version 0.2: Started monitoring at Tue
Aug
31 16:40:14 2002
...
State       ------throughput------ -----wait queue----- -----active
queue--
--
disk   r/s w/s  Kr/s Kw/s qlen res_t svc_t %ut qlen res_t svc_t %ut R
c0t3d0 26.9 8.1 108.0 90.0 0.00 0.01 0.01  0   2.01 57.31 12.56 44
Adrian detected slow net(s): Tue Aug 31 16:40:39 1999
Add more or faster nets
State Name      Ipkt/s  Ierr/s  Opkt/s  Oerr/s Coll% NoCP/s Defr/s
red   le0        598.8     0.0   541.3     0.0  0.43   0.07 100.55
Adrian detected RAM shortage (amber): Tue Aug 17 16:40:39 1999
RAM shortage
 procs        memory         page             faults           cpu
r  b  w   swap   free    pi  po  sr  rt in     sy    cs  smtx us sy id
1  1  4  155926  2535    115 340 241 37 2568   787   1317  29 49 24 28
```

The SE toolkit can monitor processes, network interfaces, disks, and more. It can even monitor Web sites. The best way to become familiar with the tool (other than downloading and using it) is to read Adrian's performance column in *SunWorld* (www.sunworld.com). One of the specific issues to read is May 1999 (www.sunworld.com/swol-05-1999), but SunWorld has a search engine so that you can easily find other references as well.

The three packages provided with the toolkit include:

1. RICHPsw, the interpreter (called SymbEL).
2. RICHPsex, the extensions package.
3. ANCrules, Adrian's rules and tools.

As the authors like to claim, all the "real work" is done in scripts. This means that you have the flexibility to modify rules if you disagree with them and the ability to create your own. If you're very clever and come up with some useful scripts, consider sending copies to the authors for distribution in their next release. As of this writing, the tool is in release 3.1.

Sun Strides in Enterprise Management

Despite the frantic pace that characterizes many system administrators' workdays, the growing size and complexity of the average network, and the availability of numerous tools to assist with the process of automating network management and monitoring, network management has not gained the foothold that one might have expected. The reason for this is, in part, price. Network management packages, such as Tivoli and Sun Net Manager, come with fairly substantial price tags. Even so, there are other reasons that we suspect are even more significant in explaining their absence on the typical network—in particular, the time and difficulty of configuring and maintaining these tools. Many companies feel that they cannot afford to train and dedicate an individual to the monitoring task, opting instead to tackle problems as they arise. As a result, the benefits anticipated from the development of automated management tools have not been manifested in the average network.

Sun has taken some large strides in overcoming this inertia with the hopes of making network monitoring and management a more common part of the average network. Two of these, the evolution of Sun Management Center and the introduction of support for Web-based enterprise management (WBEM) are the subject of this section.

Sun Management Center

If you've never heard of Sun Management Center, don't think you've fallen asleep at the console. Sun Management Center (sometimes referred to as SMC and sometimes called MC to distinguish it from Solaris Management Console) is a system management suite that evolved primarily out of Sun Enterprise SyMON. Though it has been around for a couple years or more, the tool is slowly adding features and becoming better equipped for the serious work of managing the integrated enterprise.

Sun Management Center comes with a Java GUI that can be accessed from anywhere on a network and a Web-based interface that makes it accessible from any browser (including Windows systems). A command-line interface is also now available, though the authors find it difficult to imagine that it would be nearly as effective as a GUI, because of the nature of network management.

Sun Management Center is intended to provide system administrators with the ability to manage hundreds, even thousands, of systems from a single console. Some of the tools go as far as providing photorealistic images of many types of systems and add-on components that provide diagnostics on specific hardware, such as the SunFire 15K, along with hierarchical views of system components. This type of information has, in the past, not been readily accessible to most systems administrators. One of the authors, in fact, can remember the days when the field engineer's small notebook of vital information on Sun hardware was highly coveted and completely unavailable for purchase. To make the invitation to move up to centralized network management even more enticing, Sun Management Center is free for any number of managed nodes. Some of the add-on packages such as Advanced Systems Monitoring, Premier Management Applications, and System Reliability Manager can be had for a nominal cost.

The Sun Management Center is based on open standards and is extensible, so that you can modify it to manage and monitor your applications in addition to standard systems and hardware. It can manage and monitor both hardware and operating systems, including disks, power supplies, and network statistics, and can even be extended to monitor specific applications.

Built on a traditional manager/agent model and based on SNMP, Sun Management Center distributes the processing load across the network. The management station sends requests to agents running on various devices. In other words, processing is done on the device being managed, and information is fed back to the server. Agents collect and process data locally, and can act on data and send SNMP traps. They can also run processes to correct error conditions.

Sun Management Center also provides a development environment (SDK and GUI) to allow users to create or modify modules. Though it can be used entirely on its own, it can also be used with integration modules to allow it to be integrated into Unicenter TNG, Enlighten, Halcyon, Openview, or Tivoli environments.

Some of the features of Sun Management Console are:

- Performance monitoring that enables systems administrators to identify bottlenecks
- Remote system configuration allowing consistent and reliable execution of complex configuration tasks
- Isolation of hardware and hardware faults
- Predictive failure analysis providing guidance to systems administrators on the likely failures based on statistics such as the minimum time between failures (MTBF)
- System health monitoring that incorporates a significant amount of administrative knowledge about the systems, including a rules-based health monitor that evaluates various metrics and makes suggestions for their resolution
- Log file scanning that parses log files for important information on the health of a system
- Hardware component images with pointers to specific components that help junior administrators identify which component in a system needs to be replaced
- Hierarchical component diagrams that assist system administrators in understanding the nature of a fault
- Event and alarm management, enabling system administrators to intercept important information about the status of devices in a timely manner

Software Topology

The software is organized in three tiers to provide flexibility, as depicted in Figure 9.3. The layer closest to the user is the console. In fact, this can be a number of consoles. The primary user interface is graphical. Because the interface software is written in Java, it is platform-independent. The console provides system administrators with a visual representation of the systems and other hardware they are managing. In addition, it can be customized and extended for the particular site.

The middle layer, the server layer, acts as focal point for all management and monitoring activity. It is something of a central clearing house for information on all managed devices. Communication is handled through a well-defined application programming interface (API). It communicates with agents or special processes running on each of the managed devices, receiving alarms from managed devices, and passing this information to the consoles from which the state of the network is being monitored and responding to console requests.

The agent layer, which consists of both agents and probe daemons, monitors individual components in keeping with their character. For example, an agent reporting on the state of a disk might report how full the file systems are and the level of ongoing I/O. Agents can monitor many diverse objects, including servers, desktops, storage systems, routers, and services. Agents can generate alarms or perform some corrective action based on what they detect.

If you, for example, decide that you want to monitor some aspect of a running system, such as the number of processes running at any particular time, the server layer will receive a request from the console and forward it to the appropriate agent. Meanwhile, if an agent notes an error condition on a particular device, it notifies the server of this event and the server informs the console via an alarm.

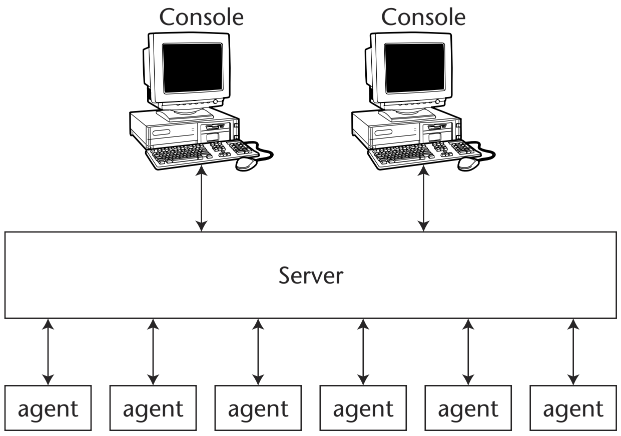


Figure 9.3 The layers of Sun Management Center.

The server layer is itself non-monolithic as well. It includes five separate components, including:

The Sun Management Center Server. This component is multithreaded and handles requests from various users at various consoles.

The Topology Manager. This component manages administrative domains as well as the topological layout of managed devices.

The Trap Handler. This component receives SNMP traps, logs traps and forwards traps to devices. It is responsible for all alarm notifications.

The Configuration Manager. This component provides security services both to the server and to the various agents.

The Event Manager. This component sends and receives event information by communicating with agents.

Each of these components is illustrated in Figure 9.4.

Sun Management Center Concepts and Components

This section introduces some of the most critical terminology associated with the Sun Management Center. In the sections that follow, we explain how these terms relate to the functionality of the software.

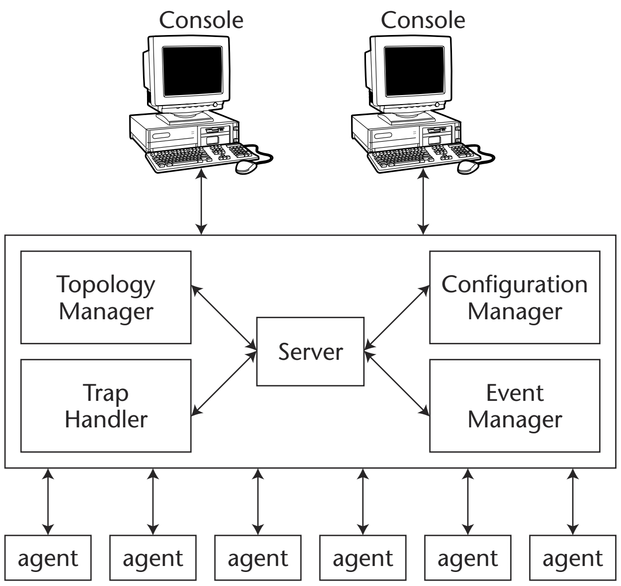


Figure 9.4 Subcomponents of the Sun Management Center Server layer.

Administrative Domains

An *administrative domain* is a collection of resources arranged hierarchically. For example, a domain might be your entire company and, within it, your divisions or departments, and within that, groups and resources. An administrative domain can be very small, containing only a handful of devices to be managed, or it can contain information pertaining to systems across an entire corporate or educational campus.

Administrative domains should be created such that they mimic and support your business needs. For example, if different groups of people manage different sets of resources, it makes sense to break these resources into separate domains.

Modules

Modules are software components that implement MIBs (discussed earlier in this chapter). Each agent may be associated with multiple modules. Each module monitors some type of resource and notifies system administrators (via their consoles) when errors occur or performance is poor.

An example of a module is the Kernel Reader, which informs you when kernel resources are being depleted.

Alarms and Rules

Systems can be monitored using alarms of severities to more accurately reflect the nature of a problem that is detected. For example, with respect to monitoring disk space, you may issue one alarm when disk usage exceeds 90 percent of capacity, a different alarm when it exceeds 95 percent, and another alarm when it exceeds 98 percent.

Though the software arrives with alarm defaults, you can modify these to your liking. Simple *rules*, like those that define when disk space alarms are to be set off, can be modified by users, while complex alarms involving multiple conditions cannot.

Autonomous Agents

Each agent works independently (or autonomously) to collect the data that it is programmed to monitor. This data is sampled on an ongoing basis. These agents can then be polled at any time through SNMP get requests. In response, they return the most current data.

Agents also "know" to check the values of data they are monitoring against thresholds that have been set for these resources. If the values meet or exceed a threshold, the agent performs the action associated with the particular alarm. Asynchronous messages sent to the server are called *traps*. The server then updates information provided to the various consoles.

Main Console Window

The Main Console Window allows you to simultaneously monitor multiple administrative domains.

Hierarchical and Topological Views

The main console window provides two panes, each of which depicts a different view of the administrative domains being monitored. The *hierarchical view* (or "Hierarchy View") provides an outline-style view that clearly depicts the hierarchical relationship between the objects. The *topological view* (or "Topology View") depicts the members of an object at that level. These views are illustrated in Figure 9.5.

View Log

The View Log button provides a way for system administrators to view informational and error messages that have been logged.

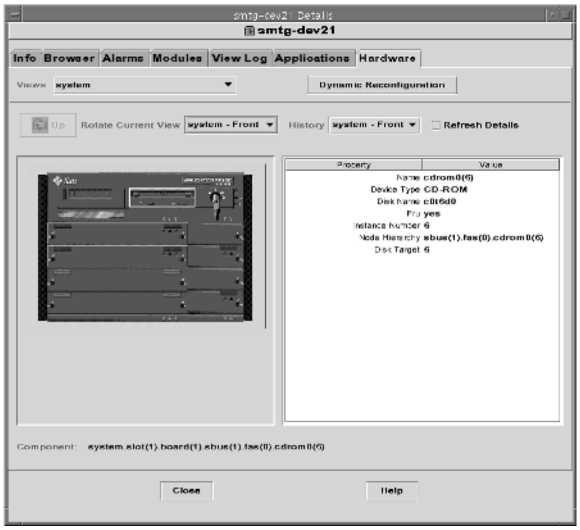
Applications

The Applications tab allows you to investigate processes that are running on managed systems. The views are updated on an ongoing basis.

Hardware

The hardware view, which was alluded to earlier in this chapter, provides photorealistic images of a system, often including views from different angles. These views help pinpoint components that might need to be replaced, making it possible for system administrators to work with unfamiliar hardware. These views are available only for some systems.

Logical views, in which the relationships between hardware components are illustrated, are available for some systems and not others.



Views: system

Rotate Current View: system - Front

History: system - Front

Property	Value
Name	cdrom0(6)
Device Type	CD-ROM
Disk Name	c0t6d0
Fru	yes
Instance Number	6
Node Hierarchy	sbus(1).fas(0).cdrom0(6)
Disk Target	6

Component: system.slot(1).board(1).sbus(1).fas(0).cdrom0(6)

Figure 9.5 Hardware (physical) view.

Graph

Two-dimensional graphs can be created from any numerical data.

Security

While the ability to monitor network resources may be generally useful, the ability to make changes to system configurations or network devices must be controlled. The security feature of the Sun Management Center allows system administrators to restrict access to managed resources by resource and user.

Software Environments

The Sun Management Center is set up so that it can easily be configured to provide two separate environments: a production environment in which your organization's critical resources are managed and monitored, and a development environment in which new modules can be developed and tested without putting production systems at risk.

Putting Sun Management Center to Work

Sun Management Center may be free but it's not a free ride. There is still a good amount of work to do to install and configure the tool. We suggest that you only consider running it on a dedicated console—an Ultra 60 or better. The software requires significant disk space and performance to run well. The User's Guide for version 3.0 is nearly 600 pages long, so don't expect to have the tool running in one afternoon.

Installation of the basic tool requires the application of appropriate patches afterwards. We found the installation to be a bit cumbersome and modifications to the `/etc/system` file require a reboot.

The Sun Management Center may end up being an important addition to a certain class of sites, especially those with a number of large servers (e.g., SunFires), where the investment is significant and the systems complex enough that any tool helping to manage them is worth its weight in gold. In other environments, its adoption is likely to be slow because of the complexity involved in setting up the software. The basic steps are:

1. Install the software.
2. Create an administrative domain.
3. Populate the domain by creating objects and/or by using the discovery manager.
4. Examine the hierarchical and topological views available through the GUI software.
5. Load and unload modules.
6. Explore modules, processes, logs, and views.
7. View property tables and graphs.
8. Create and test alarms.
9. Creating alarm conditions.
10. Establish security

By the time you get through these steps, you will probably not only have a domain to manage, but you'll be familiar with the various tools and interfaces available to you.

We recommend the use of Discovery Manager to automatically collect information on as many network devices as possible, saving yourself the work of collecting this information through other, more time-consuming means. Using Discovery Manager, you can populate your administrative domains automatically. The Discovery Manager is able to determine IP addresses, subnet addresses, hostnames and object identifiers (OIDs), and routers for various systems. You can initiate the discovery process when you first create an administrative domain with the Populate Now option or you can do this later by selecting the domain and choosing Tools ➪ Discover from the main console window.

Solaris WBEM Services

Web-Based Enterprise Management (WBEM) is a protocol that has much in common with the more familiar SNMP. Like SNMP, WBEM came about as an industry initiative to provide a way to manage systems, networks, applications, and users across multiple-vendor networks. However, it is based on a different set of standards. WBEM was designed to be compatible with SNMP, DMI, and CMIP. WBEM incorporates the following standards:

Common Information Model (CIM). A data model for describing resources to be managed.

Managed Object Format (MOF). A language which is used for defining CIM classes and instances.

Extensible Markup Language (XML). A markup language used to describe managed resources on the Web.

Whereas SNMP is the de facto standard for network management, defined by the IETF and has been in use for many years, WBEM is a relative newcomer. Designed by the Distributed Management Task Force (originally the Desktop Management Task Force), WBEM provides a unifying mechanism for describing and sharing management information, regardless of platform.

Solaris WBEM Services software and the associated developer toolkit will provide developers with the ability to create and modify system information stored in the standard CIM format. In the long run, these services will further support efforts to properly manage highly heterogeneous networks.

Summary

Monitoring your network can take all the time you have if you let it. Don't. There are far too many tools—big, expensive, and powerful as well as small, free, and flexible—that can help you chase after the details of what has gone and might go wrong.

Commercial network management systems have become very sophisticated and can provide you with near instant feedback even if you're managing a large, segmented network. Most also provide the means to manage systems and device configuration from a central station or from cooperating management stations distributed throughout your organization. Some also provide trend analysis and tools for predicting problems long before they become bottlenecks or failures.

Tools for configuring your own network management console are plentiful as well. We've covered some of our favorites in this chapter and thrown in some scripts of our own to help make sure that you spend your time looking at the more important of the data available to you.

We encourage you to continue your research into tools by reading online magazines like *SunWorld* (see www.sunworld.com) and by perusing the Internet for tools contributed by other systems administrators.

Sites such as www.sunfreeware.com and http://nce.sun.ca are great starting points for locating tools.

CHAPTER

10

Managing Performance

There are four basic reasons why performance analysis is one of the most difficult aspects of systems administration:

- The available data is overwhelming.
- There are too few rules of thumb from which to jump to easy conclusions.
- Performance analysis requires significant insight into the way system components work.
- Most system administrators only look at performance data when there is a crisis and are, therefore, unaware of how the system performs under normal conditions.

There are two main components of managing performance: performance *monitoring*, which involves collecting performance data over time and making it available graphically or otherwise, and performance *tuning*, which involves modifying the way some system component operates with the goal of removing or reducing a bottleneck.

This chapter examines the system components that need to be monitored, the commands that can help you analyze performance, and some ways in which you can tune a system for better performance.

Why You Need to Monitor Performance

The only way to fully understand performance is to know how the operating system works. If you understand the following things, you will have no problem looking deeply into performance issues and coming up with answers:

- The resources that processes use and how the scheduler works
- How memory is managed and how paging and swapping occur
- How data is read from and written to disks
- How file systems operate
- How the network interface and protocol stack operate

With this level of understanding, every measure of system performance will make sense and contribute to your picture of the system's overall health.

In the real world, however, extremely few system administrators have time to become performance experts. They are often responsible for so many systems and so many tasks that the bulk of their time is spent "firefighting." Even so, armed with a basic understanding of how systems work and focusing in on a small selection of performance statistics, it's possible to identify the cause of any system slowdown and determine what action to take.

Before we get into tools and commands used to monitor and analyze performance, let's briefly examine the concept of a performance bottleneck and how it helps add focus to performance management.

Understanding Bottlenecks

One of the frustrating truths of performance tuning is that almost anything that you do is a trade-off. Short of adding memory, replacing a disk with a faster one, and moving applications to other systems, any change you make is likely to improve performance of one aspect of a system at the cost of something else. Increasing the amount of memory that a process uses might reduce how much paging is required, but may slow down other processes competing for that memory. Increasing swap space may leave a file system with inadequate free space. At any point in time, some component ends up being the resource that cannot respond fast enough to keep up with demand; this is the definition of a performance *bottleneck*.

And the word bottleneck expresses this general problem very well. A large bottle with a long slender neck will not allow its contents to flow quickly into another container. In similar manner, any component of a system that cannot keep up with the demand placed on it slows down overall system performance. Any heavily used system is likely to have some bottleneck, some component that is used to capacity and is creating a backlog.

When a system is slow, the first thing you want to know is what is causing performance to degrade. Bottlenecks generally occur for one of the following reasons:

- Too many processes are competing for access to the CPU or one or more processes are using the CPU excessively.
- Memory is in such high demand that the system is heavily paging or swapping.
- Disks are slow or I/O is excessively heavy.
- The network interface is overloaded.

Tools and commands that provide insights into the performance of each of these basic system components—the CPU, memory, disks and network interface—are discussed in this chapter.

Performance Monitoring

Performance monitoring involves monitoring numerous aspects of a system's performance so that bottlenecks can be identified and resolved and so that future capacity can be planned with considerable insight into how a system's resources are being used today.

A Word about Performance Metrics

Before we examine each of the major potential bottlenecks and the numerous commands that can provide you with insights into how the related system components are performing, a brief glimpse into the manner in which performance metrics are collected and reported is in order.

From the time that a Solaris system boots, it collects performance data. Data structures in the kernel are updated to reflect all sorts of operating system activity. Every time a page is moved out of memory or a file is updated on disk, counters are updated to keep track of these occurrences.

Performance commands then access these kernel counters to report what is happening. Many of the same kernel counters are used by `vmstat`, `sar`, `top`, and other commands used to probe into performance.

Most of the standard Solaris performance commands have two additional characteristics in common:

- The first line of output is often a summary of the particular performance metrics since the system was booted.
- The commands can be run with both an interval and count. The *interval* specifies how many seconds of data to collect and the *count* indicates how many such intervals should be reported.

Kernel counters are updated more frequently than you might think. Some are updated many times a second. If you and someone else run the same command at the same time on the same system, you are likely to get different results—at least with many of the commands that we will look at in this chapter—because of the frequency at which the metrics change.

Monitoring Your CPU

Whether your system has one CPU or a number of CPUs, every process on your system competes with others for access to the CPU and other system resources. Access to the CPU is broken into time slices and controlled by the scheduler. Though system administrators generally think and talk in terms of processes, Solaris scheduling is actually done at the level of a thread. Because each thread acts as a separate process, the distinction may seem moot, but it is useful to keep in mind that threads are scheduled independently of each other, and yet are invisible outside of the process within which they are running.

Process Priorities

The scheduler assigns a processor to the highest priority thread from one of four run queues or *dispatch queues* (lists of processes that are not waiting for resources and are, thus, ready to run) when a processor becomes available. The thread then runs for its allotted time or prematurely gives up control of the CPU to wait for resources (e.g., data from disk or a response from a user at the keyboard).

The four dispatch queues are:

Timesharing (TS). Used for normal user processing.

Interactive (IA). Elevated in priority to accommodate the demands of X Windows.

System (SYS). Used for kernel processes.

Real-time (RT). Used for processes that require an immediate response.

Thread priorities for the timesharing class are both assigned and calculated. Though a thread generally inherits the priority of the thread that created it, it is also adjusted as a thread acquires CPU time. This prevents a single thread from monopolizing the CPU and prevents deadlocks. For example, a high-priority process might be waiting for resources tied up by a lower-priority process that is unable to run and then free the resources because of its low priority.

Process priorities can also be affected by use of the `nice` and `renice` commands, which lower the priority of a process (i.e., being nice) to allow other processes to run more often.

WARNING Not all Unix commands that report process priorities use the same numbering scheme. Some commands report priority in the "old way"; the lower the number, the higher the priority of the process. Other commands do the opposite; the higher the number, the higher the priority of the process. To avoid being confused, make sure that you know which reporting convention is being used before you interpret your process data.

Process States

In the course of its life cycle, a process moves through a series of states, as indicated in the simplified state diagram shown in Figure 10.1. When a process is first created, it is ready (that is, in the Ready state) to run and is put on the dispatch queue appropriate to its process class. In time, the scheduler dispatches the process, giving it access to the CPU (in the Run state). If the process gets to a point at which it is waiting for resources, it blocks (Blocked state). If the process runs for its allotted time slice, it is preempted and put back in the dispatch queue (Ready state). Once a process that has been blocked acquires the resources for which it has been waiting, it also moves back to the dispatch queue (Ready state). When a process is completed, it is terminated.

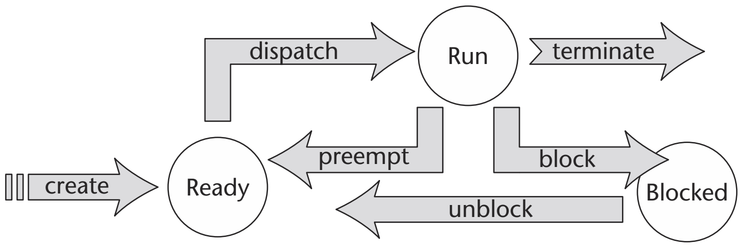


Figure 10.1 Process state changes.

The process states associated with each of these changes are explained in Table 10.1. As you shall see shortly, these process states are displayed by various `ps` commands.

Process Resources

Processes use a number of resources above and beyond the space in memory that they occupy. These resources include space in the process table, references to open files (file descriptors), and such. Performance tuning involves understanding how these resources are used and what happens when they are exhausted.

From the time a process terminates to the time that these resources are freed, the process state is set to zombie and generally appears with the process name as `<defunct>` in `ps` output. In Solaris 9, zombies can be removed and these resources freed immediately with the `preap` command.

CPU Monitoring Commands

A number of commands provide useful information on how your CPU is performing and the specific processes that are running.

Table 10.1 Process States

PROCESS STATE	DESCRIPTION	FIGURE TAG
O	Running; *on* a processor	Run
S	Sleeping; process is waiting for an event or for I/O	Blocked
R	Runnable; process is in the dispatch queue	Ready
Z	Zombie; process is terminated but resources have not yet been recovered	Not shown
T	Terminated or stopped; process has been stopped either by a signal or because it is being traced	Not shown

uptime

One of the simplest and most direct CPU monitoring commands is the `uptime` command. The `uptime` command, an example of which follows, tells you (as its name implies) how long the system has been up and running. However, it also tells you how many users are logged in and provides the 1 minute, 5 minute, and 15 minute averages of the size of the run queue.

```
# uptime
10:21am up 16 day(s), 20:23; 5 users, load
 average: 0.02, 0.02, 0.02
```

This output indicates that the particular system has an extremely light load. In each of the load averages shown, a single process was ready and waiting to run once in fifty samplings. Contrast that with this output:

```
# uptime
10:21am up 98 day(s), 09:45; 4 users, load
 average: 3.27, 2.65, 2.11
```

In this output, we can see that the system has from two to four processes ready and waiting to run, on the average, every time the size of the run queue is sampled. Where the CPU in the first sample output was idle most of the time, processes in the second are not only delayed because of competition for the processor, but also appear to be increasingly delayed.

ps

The most obvious command for monitoring what the CPU is doing is `ps`. The `ps` command provides you with a list of running processes, tells you which processes are currently running, sleeping (waiting for resources), and stopped. It will also tell you when each process began running, how much CPU time each process has accumulated, how large each process is, and quite a bit more. Though most system administrators seem to prefer the `ps -ef` command, there are numerous options available that provide additional information about running processes. We will look at several of these.

Following is some sample output from the `ps -el` command; see Table 10.2 for an explanation of the data columns:

```
# ps -el
 F S  UID   PID  PPID  C PRI NI     ADDR   SZ    WCHAN TTY  TIME CMD
19 T    0     0     0  0   0 SY fec14660    0          ?    0:00 sched
 8 S    0     1     0  0  41 20 e1b1d728  154 e1b1d94c ?    1:50 init
19 S    0     2     0  0   0 SY e1b1d008    0 fec2bc50 ?    0:01 pageout
19 S    0     3     0  1   0 SY e1b20730    0 fec7b008 ?   89:30 fsflush
 8 S 5025 19443 19437  0  40 19 e23007d0 1769 e1fc708e ?    0:01 dtwm
```

Table 10.2 ps -el Headings

HEADING	DESCRIPTION
F	Flags (historical only)
S	Process status
UID	Numeric ID of the process owner (most of those shown are owned by root)
PID	Process ID
PPID	Process ID of the parent process
C	Processor utilization for scheduling (obsolete)
PRI	Process priority
NI	Nice setting
ADDR	Address of the process in memory
SZ	Size of the process
WCHAN	Address of resource on which process is waiting
TTY	Terminal associated with process (none in this example)
TIME	Accumulated time on the CPU
CMD	Process name

We can see from this sample output that fsflush, the process that flushes (writes) all write buffers to disk periodically (usually every 30 seconds), has accumulated the most CPU time since the system booted. This is no surprise. The init process has been running as long, but accumulates little CPU time because it runs infrequently. Though fsflush should not be using more than about 5 percent of your CPU, the accumulated time reflects little more than how long it has been running. The scheduler itself (sched), pageout, and fsflush have the highest priority.

Another form of the ps command, ps -ely, displays different information. In this sample we see that the fsflush process is currently running (on the processor):

```
# ps -ely
 S   UID    PID   PPID  C PRI NI   RSS     SZ  WCHAN TTY      TIME CMD
 T     0      0      0  0   0 SY     0      0          ?      0:01 sched
 S     0      1      0  0  41 20  1152   1656      ? ?        0:34 init
 S     0      2      0  0   0 SY     0      0      ? ?        0:00 pageout
 O     0      3      0  1   0 SY     0      0        ?     2268:28
fsflush
 S     0    782      1  0  41 20  1192   1664      ? ?        0:00 sac
```

There are two size columns in this display. The first, RSS, is the resident set size, the size of the process in memory.The second of these, SZ, is the same value shown in the earlier output.

Since Solaris 2.6, a -o option is also available to allow the user to customize the `ps` output that he wants to see. A command of the form `ps -o s`, for example, will display the process state of each process while `ps -o nlwp` would display the number of threads associated with each process. Combining these display options with other parameters allows you to extract meaningful but succinct output. Sample output is shown below. The statistics that can be requested with the `ps -o` command are shown in Table 10.3.

```
# ps -ef -o nlwp -o args
NLWP COMMAND
   1 sched
   1 /etc/init -
   1 pageout
   1 fsflush
   1 /usr/lib/saf/sac -t 300
   1 /usr/openwin/bin/Xsun :0 -nobanner -auth /var/dt
   7 devfsadmd
  11 /usr/lib/sysevent/syseventd
   5 /usr/lib/sysevent/syseventconfd
... and so on

# ps -ef -o s -o args
S COMMAND
T sched
S /etc/init -
S pageout
S fsflush
S /usr/lib/saf/sac -t 300
... and so on
```

Table 10.3 ps -o Options

OPTION	DESCRIPTION
ruser	Real user ID associated with the process (text)
uid	Effective user ID (numeric)
ruid	Real user ID (numeric)
group	Effective group ID
rgroup	Real group ID
gid	Effective group ID (numeric)
rgid	Real group ID (numeric)
pid	Process ID

Table 10.3 *(Continued)*

OPTION	DESCRIPTION
ppid	Parent process ID
pgid	Process group ID
sid	Process ID of session leader
lwp	Lightweight process ID (one line per lwp)
nlwp	Number of light weight processes
psr	Number of processors to which the process is bound
pcpu	Ratio of CPU time used recently to CPU time available
osz	Total size of the process in memory (pages)
vsz	Total size of process in virtual memory (kilobytes)
rss	Resident set size (kilobytes)
pmem	Ration of rss to physical memory
class	Scheduling class
nice	System scheduling priority
pri	Priority of process (higher value represent higher priority)
opri	Old priority values (lower numbers represent higher priority)
s	State of process
f	Process state flags
etime	Elapsed time since process start
time	CPU time
stime	Start time or date
tty	Controlling terminal
comm	Name of command
args	Command with arguments
addr	Memory address (beginning) of the process
wchan	Address of resource for which the process is waiting while sleeping

top

The `top` command provides a dynamic view of process status by displaying a periodically updated list of those processes that are making the heaviest use of the CPU. Though `top` is not included in the Solaris distribution, it has long been a favorite of system administrators and is available in package form from www.sunfreeware.com.

```
last pid: 27311;  load averages:  0.03,  0.03,  0.04
        10:53:26
113 processes: 112 sleeping, 1 on cpu
CPU states: 98.1% idle,  0.9% user,  0.7% kernel,  0.3% iowait,  0.0%
swap
Memory: 6144M real, 92M free, 298M swap in use, 6736M swap free

   PID USERNAME THR PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
 22737 oracle     1  38    0  122M  107M sleep    1:53  0.31% oracle

 14866 root      32  59    0   40M   19M sleep   88:31  0.09% eas
 27303 root       1  58    0 2656K 1800K cpu/0    0:00  0.08% top
 16918 root      12  58    0   38M   17M sleep   61:13  0.02% msgproc
   696 root       9  59    0   12M 9072K sleep   58:54  0.01% jre
 27243 root       1  33    0 1048K  696K sleep    0:00  0.01% sh
 14865 root      10  53    0   39M   12M sleep   34:01  0.01%
rmiregistry
 12437 oracle     1  58    0   46M   22M sleep    6:27  0.01% oracle
```

In this display, we can see that oracle is the top CPU user but, at .31 percent, it can hardly be accused of being a resource hog. In fact, the presence of the `top` process in the `top` output is itself a good indicator that we are not looking at a system with a busy CPU. Notice the low load averages on the top line. Then, move down three lines and notice that the CPU is reported to be 98.1 percent idle.

The fields in the `top` display are explained in Table 10.4.

Table 10.4 top Headings

HEADING	DESCRIPTION
PID	Process ID
USERNAME	Username of the process owner (most of those shown are owned by root)
THR	Number of executable threads in the process
PRI	Process priority
NICE	Nice value (lowers priority)
SIZE	Size of the process
RES	Resident set size (size of the process in memory)
STATE	Process status
TIME	Accumulated clock time
CPU	Accumulated CPU time
COMMAND	Command name

The other statistics on the CPU states line report the amount of time that the CPU spent servicing user processes, the amount of time the CPU spent on system (kernel) processes, the amount of time the CPU was idle while processes were waiting for I/O (some commands do not break out this statistic but report it as CPU idle), and the amount of time the CPU spent on swapping activity.

Were the output to look like the following example instead, we would be far more concerned. In this output, we have high load averages and a CPU that seems to be busy much of the time (though nearly 50 percent of this `iowait`).

```
% top
 last pid: 23085; load averages:  4.72,  5.30,  3.73

       09:31:02
 842 processes: 224 sleeping, 5 running, 2 on cpu
 CPU states: 0.0% idle, 30.5% user, 22.9% kernel, 46.7% iowait, 0.0%
swap
```

`top` also provides a brief report on memory use. We can see from the numbers provided in the first sample of `top` output that some memory is free and that swap space is only lightly used.

prstat

The `prstat` command (which stands for *process status*), first introduced in Solaris 8, provides output that is similar to both the `ps` output and `top` commands. Like `top`, `prstat` reports processes in the order defined by their CPU usage.

```
$ prstat
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 14805 shs      1028K  712K sleep   43    0   0:00.00 0.4% sh/1
 14810 shs      1304K 1068K cpu1    52    0   0:00.00 0.4% prstat/1
 12082 root       48M   26M sleep   58    0   0:00.02 0.2% java/23
 14807 root     1628K  972K sleep   38    0   0:00.00 0.1% rquotad/1
 14803 root     1732K 1096K sleep   51    0   0:00.00 0.1% in.telnetd/1
...
Total: 68 processes, 219 lwps, load averages: 0.03, 0.01, 0.02
```

The `prstat` command also continues to run, updating its display until you hit Ctrl+C to stop the output.

`prstat` contributes a count of light weight processes (threads) on each line. The rest of the columns are nearly identical to the data reported by `top`.

The fields in the `prstat` command are explained in Table 10.5.

sar

Another command that reports CPU utilization is the `sar -u` command. The `sar` command is discussed later in this chapter.

Table 10.5 prstat Headings

HEADING	DESCRIPTION
PID	Process ID
USERNAME	Login name of the process owner (most of those shown are owned by root)
SIZE	Total size of the process
RSS	Resident set size (size of the process in memory)
STATE	Process status (cpu#, sleep, run, zombie,stop)
PRI	Process priority
NICE	Nice value (lowers priority)
TIME	Accumulated clock time
CPU	Accumulated CPU time
COMMAND	Command name

proc tools

A wealth of information on processes is also available through the proc tools, a set of commands that provide information on running processes and the `/proc` file system, a pseudo-file-system-like interface into such things as the memory used by a process and the files that it has open. See the *The /proc Commands* section later in this chapter for more information on proc tools.

vmstat

The `vmstat` command also provides considerable information on processes. This should not come as a surprise since the state of memory is only relevant with respect to processes. The `vmstat` command is discussed in the *Monitoring Memory* section later in this chapter.

lsof

The lsof (list open files) utility lists files that have been opened by processes on your system. As innocuous as this may sound, this command has been referred to by some as "the Swiss army knife of intrusion detection". We will attempt to explain why we feel this label is warranted and present you with some interesting uses for the `lsof` command. The `lsof` command is not part of the Solaris distribution, but a package for Sparc or source can be downloaded from www.sunfreeware.com or from the following FTP site: ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/.

If you run the lsof command with no arguments at all, you will get a lot of output—a list of all files opened by all applications. You can pare this output down to something more likely to suit your needs by selecting an option. For example, using the -p option will cause the tool to list only those open files associated with the specified process. The output shown below was seriously truncated because there was so much of it. Obviously, this particular Java application uses a lot of jar files as well as shared libraries:

```
# lsof -p 5678
COMMAND  PID USER   FD   TYPE        DEVICE SIZE/OFF   NODE NAME
java    5678 root  txt   VREG         32,83  1936596  55926
/opt/app/lib/classes.zip
java    5678 root  txt   VREG         32,83   754041  55937
/opt/app/lib/generic.jar
java    5678 root  txt   VREG         32,75    12644  11992
/usr/lib/libdoor.so.1
java    5678 root  txt   VREG         32,83   695539  55936
/opt/app/lib/xml4j.jar
```

We also notice some log files, such as this one:

```
8595 366434 /opt/JRun/logs/app-err.log

java    5678 root    3u  VCHR         13,12      0t0 194347
/devices/pseudo/mm@0:zero
```

and some network connections, such as these:

```
java    5678 root    9u  IPv4 0x30008489a70      0t0    TCP
localhost:50009 (LISTEN)
java    5678 root   23u  IPv4 0x30008d05d00  0t25463    TCP
rollerdisco:45743->rollerdisco:1521 (CLOSE_WAIT)
java    5678 root   24u  IPv4 0x30001a8d310 0t241571    TCP
rollerdisco:45744->rollerdisco:1521 (CLOSE_WAIT)
```

To show open files for all TCP connections, you would use the lsof -iTCP command. Again, the output shown below is truncated. Notice the similarity between this output and that of netstat (which is discussed later in this chapter):

```
# lsof -iTCP
COMMAND PID  USER  FD   TYPE        DEVICE  SIZE/OFF NODE NAME
inetd   178  root  11u  IPv6 0x30001eadbe8    0t0  TCP *:ftp (LISTEN)
inetd   178  root  12u  IPv6 0x30001ead6e8    0t0  TCP *:telnet (LISTEN)
inetd   178  root  14u  IPv4 0x30001ead468    0t0  TCP *:shell (LISTEN)
inetd   178  root  48u  IPv4 0x30001ef46b8    0t0  TCP *:32777 (LISTEN)
...
```

To show open files for a particular IP address, you tack the IP address onto the command, like this:

```
# lsof -iTCP@10.4.4.49
COMMAND   PID USER   FD   TYPE        DEVICE SIZE/OFF NODE NAME
sshd     1736 root    7u  IPv4 0x300029a41b0  0t21210  TCP db4500b:22-
>henrystockers:1964 (ESTABLISHED)
sshd    27992 root    7u  IPv4 0x30003ad2420  0t30122  TCP db4500b:22-
>henrystockers:1932 (ESTABLISHED)
```

To show the files opened by a particular user, you would use this command:

```
lsof -u username
```

To get the name of the process that has a particular file open, you would use this command:

```
# lsof /usr/java/lib/rt.jar
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF  NODE NAME
java      5564 root  txt   VREG  32,75 13839483 23275
/usr/j2re1_3_1_03/lib/rt.jar
rmiregist 5696 root  txt   VREG  32,75 13839483 232755
/usr/j2re1_3_1_03/lib/rt.ja
java      5726 root  txt   VREG  32,75 13839483 23275
/usr/j2re1_3_1_03/lib/rt.jar
java      5867 root  txt   VREG  32,75 13839483 23275
/usr/j2re1_3_1_03/lib/rt.jar
```

The `lsof` command has similarities to some of the proc commands (e.g., `pfiles`) and with the `fuser` command, but has a simplicity and versatility that makes it extremely handy for troubleshooting. We recommend it highly.

Making Use of /proc

The `/proc` file system is an interesting and very useful "file system." Though its contents do not correspond to files on disk, `/proc` appears to the user as a regular file system. Users can `cd` into `/proc` as with any directory for which they have execute permission. They can list its contents with `ls` or `ls -l`.

What the contents of the `/proc` file system represent are interfaces to kernel structures and running processes. By providing an interface that looks like a file system, `/proc` simplifies the work involved in accessing information available about these processes. For example, before the advent of `/proc`, programs that used information about kernel structures (e.g., the `ps` command, which reads and displays the process table) had to read kernel memory. With the `/proc` file system, these structures are more readily available. This is useful not only to the OS developers, but to any programmers or sysadmins needing access to this type of information.

Although /proc may not be every sysadmin's favorite place to poke around, it offers advantages for debugging. With images of running processes as readily available as files, they are much easier to analyze. Solaris provides a set of tools to facilitate access even further—the commands in /usr/proc/bin, which are discussed in the next section.

When you examine the contents of /proc, the first thing you'll likely notice is that the directory names correspond to the process IDs of running processes. The init process, for example, will show up as a directory named 1. Within each process directory are a number of files or subdirectories that represent different aspects of the corresponding process.

```
spoon% cd /proc
spoon% ls
0      1129   1281   167    2      22416  23334  27148  28524  3529
362
1      11352  124    126    20201  22882  23508  27153  28535  372
436
10382  11353  13403  187    20281  230    23509  27159  28572  373
467
... and so on
```

If you were to compare this list with that generated by a ps -ef command, you would notice the matching process IDs.

```
spoon% ps -ef | head -11
     UID   PID  PPID  C    STIME TTY       TIME CMD
    root     0     0  0   Mar 11 ?         0:02 sched
    root     1     0  0   Mar 11 ?        76:54 /etc/init -
    root     2     0  0   Mar 11 ?         4:55 pageout
    root     3     0  0   Mar 11 ?        247:55 fsflush
  nobody  3529   349  0 15:18:44 ?         0:01 /usr/sbin/httpd-apache -
f /etc/httpd-apache.conf
    root   362     1  0   Mar 11 ?         0:11 /usr/lib/saf/sac -t 300
    root   124     1  0   Mar 11 ?        90:48 /usr/sbin/rpcbind
    root   126     1  0   Mar 11 ?         0:06 /usr/sbin/keyserv
    root   187     1  0   Mar 11 ?         3:05 /usr/sbin/cron
  daemon   166     1  0   Mar 11 ?         0:09 /usr/lib/nfs/statd
```

Each process is actually represented by a directory that holds a number of files and other directories corresponding to different portions and attributes of the process. Note the various structures associated with a single process in the following example. The owner and group reflect those of the person running the process. Read the man page on /proc for additional information.

```
spoon% ls -l 23308
-rw-------   1 nici     staff   1323008 May 31 12:02 as
-r--------   1 nici     staff       152 May 31 12:02 auxv
-r--------   1 nici     staff        40 May 31 12:02 cred
```

```
--w-------   1 nici     staff          0 May 31 12:02 ctl
lr-x------   1 nici     staff          0 May 31 12:02 cwd ->
dr-x------   2 nici     staff       1184 May 31 12:02 fd
-r--r--r--   1 nici     staff        120 May 31 12:02 lpsinfo
-r--------   1 nici     staff        912 May 31 12:02 lstatus
-r--r--r--   1 nici     staff        536 May 31 12:02 lusage
dr-xr-xr-x   3 nici     staff         48 May 31 12:02 lwp
-r--------   1 nici     staff       1536 May 31 12:02 map
dr-x------   2 nici     staff        288 May 31 12:02 object
-r--------   1 nici     staff       1952 May 31 12:02 pagedata
-r--r--r--   1 nici     staff        336 May 31 12:02 psinfo
-r--------   1 nici     staff       1536 May 31 12:02 rmap
lr-x------   1 nici     staff          0 May 31 12:02 root ->
-r--------   1 nici     staff       1440 May 31 12:02 sigact
-r--------   1 nici     staff       1232 May 31 12:02 status
-r--r--r--   1 nici     staff        256 May 31 12:02 usage
-r--------   1 nici     staff          0 May 31 12:02 watch
-r--------   1 nici     staff       2432 May 31 12:02 xmap
```

The /proc Commands

The set of commands that facilitate use of the information stored in a /proc file system, sometime referred to as the *proc tools,* are discussed in the following sections.

pargs

New with Solaris 9, the pargs command displays a full list of the arguments that were used when a process was initiated, as shown here:

```
# pargs `pgrep ttymon`
672: /usr/lib/saf/ttymon -g -h -p system-name console login:
-T sun -d /dev/console -l
argv[0]: /usr/lib/saf/ttymon
argv[1]: -g
argv[2]: -h
argv[3]: -p
argv[4]: system-name console login:
argv[5]: -T
argv[6]: sun
argv[7]: -d
argv[8]: /dev/console
argv[9]: -l
argv[10]: console
argv[11]: -m
argv[12]: ldterm,ttcompat
612: /usr/lib/saf/ttymon
argv[0]: /usr/lib/saf/ttymon
```

pcred

The pcred command prints the effective, real, and saved UID and GID of a running process. The command pcred 1234, as shown here, shows that process 1234 has UID of 111 and GID of 11:

```
solaris% pcred 1234
1234:  e/r/suid=111  e/r/sgid=11
```

pfiles

The pfiles command lists open files associated with the specified process along with any file limits the process may have imposed. Each open file is described by a number of values, including the inode number, major and minor device numbers of the partition in which the file is stored, the UID, the GID, and permissions. Although the identity of each file may not be easily apparent from the display, any particular file can be identified by using a combination of the device numbers and inode number, as we shall explain shortly.

```
spoon% pfiles 28555
28555:  -csh
  Current rlimit: 64 file descriptors
   0: S_IFCHR mode:0666 dev:32,24 ino:127001 uid:0 gid:3 rdev:13,2
      O_RDONLY|O_LARGEFILE
   1: S_IFCHR mode:0666 dev:32,24 ino:127001 uid:0 gid:3 rdev:13,2
      O_RDONLY|O_LARGEFILE
   2: S_IFCHR mode:0666 dev:32,24 ino:127001 uid:0 gid:3 rdev:13,2
      O_RDONLY|O_LARGEFILE
   3: S_IFDOOR mode:0444 dev:163,0 ino:59379 uid:0 gid:0 size:0
      O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[208]
```

In the example shown, the process is using nine file handles, but only three actual files. This is evident from looking at a combination of the device numbers (e.g., 32,24) and the inode number (e.g., 127001). Because the same inode number can be used in different file systems for different files, only the combination of these values is sufficient to identify a file.

pflags

The pflags command displays the status of the process, as shown in the following example:

```
spoon% pflags 28555
28555:  -csh
        data model = _ILP32  flags = PR_ORPHAN
  /1:   flags = PR_PCINVAL|PR_ASLEEP [ sigsuspend(0xeffff878) ]
  sigmask = 0x00000002,0x00000000
```

pgrep

The pgrep command searches for processes by name using the /proc interface. Many sysadmins used to using commands like `ps -ef | grep sendmail` will come to appreciate the simplification of `pgrep sendmail`.

pkill

The pkill command sends a signal to the process. Like the pgrep command, pkill sends signals to the process named on the command line. Sysadmins familiar with the killbyname script will find the operation of this command familiar. Replacing command sequences of `ps -ef | grep inetd` and `kill -HUP 101` with pkill (e.g., `pkill -HUP inetd`) saves time and simplifies scripting as well.

pldd

The pldd command lists the dynamic libraries used with the process.

pmap

The pmap command displays the process's address space with memory segment sizes, as shown in the following example output:

```
spoon% pmap 28555
28555:  -csh
00010000    140K read/exec         /usr/bin/csh
00042000     12K read/write/exec   /usr/bin/csh
00045000     68K read/write/exec     [ heap ]
EF600000    648K read/exec         /usr/lib/libc.so.1
...
 total     1300K
```

prun

The prun command sets processes running and is the opposite of the pstop command.

psig

The psig command lists the signal actions of the process.

pstack

The pstack command shows the stack trace for each process thread.

pstop

The pstop command stops processes in a manner similar to the default kill command (i.e., the kill command without an argument). It is the opposite of the prun command.

ptime

The ptime command displays timing information for processes—real, user, and system—as the following sample output illustrates:

```
spoon% ptime 28555
real        0.024
user        0.001
sys         0.017
```

ptree

The `ptree` command prints the process's tree, containing the process IDs or users:

```
spoon% ptree 28555
158   /usr/sbin/inetd -s
  28553 in.telnetd
    28555 -csh
      28586 ptree 28555
```

pwait

The `pwait` command waits for the process to terminate.

pwdx

The `pwdx` command prints the current working directory of the specified process, as shown in the following example:

```
spoon% pwdx 28555
28555:  /home/nici
```

CPU States

The way to think about the CPU statistics is this: Every time the CPU is examined (i.e., 100 times a second), the data structures reflecting how the CPU is spending its time are updated. If the CPU is quiescent, *idle* is incremented. If the CPU is busy with a user process, *user* (also referred to as "usr") is incremented. If the CPU is handling an interrupt or running the clock handler, *kernel* (in some commands, this is "sys") is incremented. If the CPU is running the idle thread, *idle* or io*wait* (in some commands, this is "wt or "wio"") is incremented depending on whether a separate waitio value is zero. It is no surprise then that some performance commands include wait I/O as part of idle statistic while others break it out.

If the kernel statistic is consistently high, you might gain some advantage by tuning the operating system. If, on the other hand, the CPU is spending the bulk of its time on user processes or disk and tape I/O (i.e., iowait), you should turn your attention to user processes or consider adding memory to the system.

Monitoring Multiple-CPU Systems

On any system with multiple CPUs, you need to know whether the statistics report the counters for each of the CPUs separately or combine them. The `prstat` command, for example, not only reports the state of each process, but reports the processor that a running process is using. In this line from the prstat output shown earlier, we can see that the mpstat command is running on cpu1:

```
14810 shs      1304K 1068K cpu1    52    0   0:00.00 0.4% prstat/1
```

sar averages the counters for all processors.

mpstat

The mpstat command is the only process that breaks out important counters for each of the CPUs. In the output shown below, mpstat is being used on a six-CPU system.

```
# mpstat
CPU minf mjf xcal  intr ithr csw icsw migr smtx srw syscl usr sys wt idl
  0    9   0   45   103  101 250   16    8    3   0   236   1   2  1  96
  1   15   0  566   400  200  79    2    4    4   0  1121   1   1  1  97
  4    9   0   42   101  100 214   14    7    4   0   416   1   1  1  97
  5   10   0   44   139  138 212   13    7    4   0   813   1   2  1  97
  8    9   0   41   114  113 209   12    6    4   0   363   1   1  1  97
  9    8   0   49   105  104 220    9    7    4   0   729   1   1  1  97
```

In this sample output, we see that all six CPUs are similarly idle. The mpstat command can tell you how each CPU is spending its time and whether the workload is being distributed equally among the processors.

The column headings in the mpstat command are described in Table 10.6.

Table 10.6 mpstat Headings

HEADING	DESCRIPTION
CPU	CPU number
minf	Minor faults
mjf	Major faults
xcal	Interprocessor cross calls
intr	Interrupts
ithr	Interrupts as threads
csw	Context switches
icsw	Involuntary context switches
migr	Thread migrations (to another processor)
smtx	Spins on mutexes
srw	Spins on reader/writer locks

Table 10.6 *(Continued)*

HEADING	DESCRIPTION
syscl	System calls
usr	Percentage of time spent on user processes
sys	Percentage of time spent on system processes (e.g., system calls)
wt	Percentage of time in I/O wait
idl	Percentage of time idle

Without arguments, the `mpstat` command provides summary statistics from the time the system was booted. With a single argument (e.g., `mpstat 20`), you would get the summary line followed by a report indicating the activity in the specified number of seconds. With two arguments (e.g., `mpstat 20 10`), you would get the summary report followed by a report for each of the requested intervals.

In the preceding `mpstat` output, one line of the output stands out as being different from the other five. Though the activity on `cpu1` illustrates that this system had more interprocessor cross-calls, interrupts, interrupts as threads, and system calls, this processor also handled fewer context switches and was, overall, no busier than the rest.

Monitoring Memory

Memory is used by every process on a system. In fact, at any point in time, portions of many different processes will be sitting in memory together. Other portions of these processes will have been written to a special disk file.

Memory is divided into segments referred to as *pages* (4 or 8 KB in size, depending on the particular hardware). These pages are numbered and moved in and out of memory as needed to keep the currently active portion of the various processes positioned for use. Virtual memory is a set of algorithms that allows the apparent memory to be much larger than physical memory, thus making it possible for larger processes to run and more processes to run at the same time.

When a portion of a process isn't needed, it is written out to a paging file. It may be loaded many times and written to the paging file only once. As long as the page doesn't change, there is no need to write it out again.

If the system determines that a page needed by a process is not in memory, it reads it into memory. The discovery that the page is not already in memory is called a *page fault*, though this is a normal and frequent occurrence. When space in memory needs to be cleared so that pages can be read, the paging system (i.e., the *pagedaemon*) provides statistics that allow the system to determine which of the currently resident pages were least recently used. By removing these particular pages, the paging system makes it more likely that the pages needed next will already be in memory.

The process of paging is normal and is designed to be as efficient as possible. When paging is excessive, however, the system can resort to a different kind of memory management. Instead of moving individual pages or portions of processes to the paging file, the system moves entire processes out of memory and writes them to the system swap space. This process is called *swapping*.

Swapping is similar to paging in that the contents of memory are written to disk. Swapping, however, is more severe because entire processes are moved out of memory until the paging level decreases and are then gradually reloaded.

Monitoring the use of memory can, therefore, involve several activities:

- Looking at how fully memory is used and how much memory is free
- Determining how much swap space is in use
- Examining the level of paging that is occurring

Some of the commands already covered provide a snapshot of memory. For example, the `top` command showed us this:

```
Memory: 6144M real, 92M free, 298M swap in use, 6736M swap free
```

This output indicates that memory is not completely used and that swap space is used minimally.

The `vmstat` command tells us considerably more about what's going on with memory than `top`, as shown here. We suggest `vmstat 30` that samples virtual memory every 30 seconds. Ignore the first line of output; this is summary data.

```
# vmstat 30 5
 procs     memory              page             disk           faults
cpu
 r b w   swap  free   re  mf  pi  po fr de sr s6  s9 s1 --   in   sy   cs us
sy id
 0 0 0 5200232 1288008 600 5891 22 2 2 0 0  0 10 32  0  907 6919 2064 23
35 42
 0 0 0 5192264 1352920 1442 4438 3 3 3 0 0  0  1  3  0  759 6274 1706 57
28 15
 0 0 0 5192112 1423288 1330 5015 681 4 4 0 0 0 9 72  0 1071 6948 2041 51
34 15
 0 1 0 5192032 1414632 632 4819 870 3 3 0 0 0 28 119 0 1359 6760 2492 31
32 37
 0 1 0 5191040 1403768 519 4806 185 3 3 0 0 0 26 93  0 1272 6666 2424 26
31 43
```

This output allows you to focus in on some numbers that tell you quite a bit about how memory is working. First, you can ignore the rightmost three columns of data; these three numbers provide another synopsis of how the CPU is spending its time, this time with iowait subsumed by the idle statistic.

The leftmost three columns display the number of processes in the run queue ("r"), blocked for resources ("b"), and runnable but swapped out ("w"). For the largely idle system on which this output was generated, it is not surprising that all three of these numbers are zeroes.

Next, we see that the free swap and free memory statistics (both reported in kilobytes) indicate that memory is not stressed.

Two of the most useful statistics for determining whether memory is stressed are included under the "page" heading. The pageout (po) and scan rate (sr) statistics indicate how many kilobytes were paged out in the sample period and the number of pages scanned while the system was looking for pages to move out of memory. Numbers consistently above zero for these figures warrant attention. Scan rates above 200 suggest that additional memory should be added to the system. Scan rates exceeding 320 can lead to *thrashing*, a situation in which paging becomes so excessive that no real work is accomplished. During thrashing, the page scanner itself may end up using as much as 80 percent of processor time.

The statistics reported by the `vmstat` command are explained in Table 10.7.

Table 10.7 vmstat Headings

HEADING	DESCRIPTION
r	Processes in run queue
b	Processes blocked for resources
w	Processes that are runnable but swapped out
swap	Amount of available swap space
free	Size of the free list in kilobytes
re	Page reclaims from the free list
mf	Minor faults (address space and hardware translation faults)
pi	Kilobytes paged out
po	Kilobytes paged in
fr	Kilobytes freed
de	Anticipated short-term memory shortfall (kilobytes)
sr	Pages scanned by the clock algorithm
disk (s0, s1 etc.)	Disk I/O operations per second for 1-4 disks
in	Interrupts
sy	System calls
cs	Context switched
us	User time
sy	System time
id	Idle time

With a -s argument, vmstat summarizes statistics since the system was booted and includes additional information on swapping activity. The following output shows a sample of the report that is provided:

```
vmstat -s
        0 swap ins
        0 swap outs
        0 pages swapped in
        0 pages swapped out
3139721419 total address trans. faults taken
   278417 page ins
    73615 page outs
  1517746 pages paged in
   136999 pages paged out
320176029 total reclaims
320175468 reclaims from free list
        0 micro (hat) faults
3139721419 minor (as) faults
   236515 major faults
373778523 copy-on-write faults
201239008 zero fill page faults
        0 pages examined by the clock daemon
        0 revolutions of the clock hand
   136425 pages freed by the clock daemon
 14893422 forks
    23033 vforks
  9103362 execs
1100499399 cpu context switches
536995432 device interrupts
3676945603 traps
3687498648 system calls
2311989495 total name lookups (cache hits 99%)
 24880801 user   cpu
 37160994 system cpu
 34568051 idle   cpu
  9969275 wait   cpu
```

The vmstat -p command reports paging activity for applications, data, and file systems. Sample output is shown here:

```
# vmstat -p
 memory         page        executable    anonymous      filesystem
swap  free       re  mf   fr de  sr epi   epo  epf  api  apo  apf  fpi fpo  fpf
5200224 1288064 600 5891 2 0  0    0    0    0    0    0    0   22   2    2
```

NOTE In Solaris 8 and later, due to changes in the algorithm used to remove pages from memory, some `vmstat` statistics will be higher or lower than on pre-Solaris 8 systems. These changes include:

- **The page scan rate will be higher.**
- **Free memory will be larger.**
- **Scan rates will be close to zero unless there is a systemwide memory shortage.**

The cylindrical page cache provides a separate cache for file system data, removing the earlier competition between file system caching and virtual memory.

Page scanning (i.e., the scan rate) is also reported by the `sar -g` command, a sample of which is shown here:

```
# sar -g 10

SunOS rollerdisco 5.8 Generic_108528-05 sun4u    05/20/02

18:25:35  pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
18:25:45     0.20     0.40     0.40     0.00     0.00
```

Disk activity to the swap device can also be seen with the `iostat -xPnce` command, available in Solaris 2.6 and later. Sample output is shown here:

```
# iostat -xPnce
     cpu
 us sy wt id
 15 27  6 52
                            extended device statistics       ---- errors
---
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b s/w h/w trn
tot device
    0.0    0.0    0.0    0.0  0.0  0.0    1.5   58.0   0   0   0   0   0
0 c0t10d0s0
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    5.1   0   0   0   0   0
0 c0t10d0s1
    0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0   0   0   0
0 c0t10d0s2
```

This command is explained further in the *iostat* section later in this chapter.

Information on the use of shared memory is available through the `ipcs -mb` command.

```
# ipcs -mb
IPC status from <running system> as of Sun Jun  2 18:19:50 EDT 2002
T        ID      KEY        MODE        OWNER    GROUP      SEGSZ
Shared Memory:
m        0   0x50080653 --rw-r--r--     root     root       4
```

An obvious command for looking at use of swapping is the swap command. The `swap -s` command reports on the amount of swap space that is:

Allocated. Currently allocated.

Reserved. Not currently allocated but reserved.

Used. The total of the allocated and reserved spaces.

Available. Neither allocated nor reserved (free).

The `swap` command shown here illustrates these values:

```
# swap -s
total: 302332k bytes allocated + 121808k reserved = 424140k used,
3032084k available
```

The `swap` command is also used to list swap partitions and to add and remove swap partitions or files. Sample output from the `swap -l` command is shown here:

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t1d0s1   29,129      8 2050120 2050120
```

The `mkfile` and `swap -a` commands, for example, can be used to add swap space to a system if the system runs short of virtual memory. The following two commands would create a swap file and add it to the swap space on a system (you would need to add the swap file to the `/etc/vfstab` file to add the file to the swap space permanently).

```
# mkfile 100m /opt/swapfile
# swap -a /opt/swapfile
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t1d0s1   29,129      8 2050120 2050120
/opt/swapfile          -        8 204792  204792
```

Monitoring Your Disks

Disk I/O is often a suspect when a system's performance degrades, because disks tend to be the slowest component on a computer. In addition, disks are not merely used to house file systems, they are also used to support paging and swapping. Therefore, the speed of system disks can be a major performance bottleneck.

The major performance measurement associated with disks is called *throughput*. The three factors that influence throughput from a hardware standpoint are:

Seek latency. The time that it takes the disk head to position itself over the required cylinder.

Rotational latency. The time it then takes the block to move (or rotate) under the head; keep in mind that the disk platter is spinning thousands of times each minute.

Data transfer rate. The time that it takes for the data to be delivered to the system. Actual measurements depend not only on the disk hardware, but also on the read patterns requested by the system.

iostat

The primary command for monitoring disk performance is `iostat`. Like the `vmstat` command, `iostat` first reports a line of data that summarizes activity since the system was booted. In this first `iostat` command, we are asking for 5-second samples of disk activity:

```
# iostat 5
      tty          md0           md3           md4          md10
cpu
 tin tout kps tps serv  kps tps serv  kps tps serv  kps tps serv  us sy
wt id
   0   38   3   0   27    1   0   18    1   0   12    2   0   19   1  1
1 97
   0   47   0   0    0    0   0    0    0   0    0    0   0    0   0  1
0 99
```

The iostat command reports on terminal I/O, tape I/O, and CPU activity as well as disk I/O. The columns in this display are explained in Table 10.8.

The third through fifth columns are repeated for up to four disks.

If you want to see only disk activity, try a command of the form `iostat -d 10 10`, as shown here:

```
# iostat -d 10 10
    sd6           sd9           sd10          nfs1
kps tps serv  kps tps serv  kps tps serv  kps tps serv
  0   0   19   34   3   11  155  19   22    0   0    0
  0   0    0    2   0    9    6   1   17    0   0    0
  0   0    0    2   0    9    6   1   15    0   0    0
  0   0    0    5   1   11   13   2   20    0   0    0
  0   0    0    4   0    9    6   1   19    0   0    0
```

Table 10.8 iostat Headings

HEADING	DESCRIPTION
Tin	Total characters in per second
Tout	Total characters out per second
Kps	Kilobytes transferred per second
Tps	Transfer operations per second
Serv	Average service time in milliseconds
Us	User time
Sy	System (kernel) time
Wt	Iowait
Id	Idle time

iostat allows you to determine, among other things, whether the I/O between your disks is more or less balanced. If one or more of your disks use significantly more than the others, overall disk throughput may be degraded. This output also allows you to see how quickly disks respond to requests. Large service times may be seen with slowed or more fragmented disks.

If you have more than four disks or want to see additional data, use the iostat -x command. This command provides extended statistics on disk I/O. Following is sample output for the the iostat -x command; see Table 10.9 for definitions of each of the data columns.

```
# iostat -x
                  extended device statistics
device       r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
sd6          0.0    0.0    0.0    0.0  0.0  0.0   18.7   0   0
sd9          0.1    3.4    4.1   29.4  0.0  0.0   10.8   0   3
sd10         0.0   18.7    4.0  151.1  0.0  0.4   22.4   0  15
nfs1         0.0    0.0    0.0    0.0  0.0  0.0    0.2   0   0
```

The disk I/O statistics that we consider the most telling are service times (svc_t) and disk busy percentages (%b). These two numbers will tell you whether disks are slow to respond and/or overly busy. Service times exceeding 50 milliseconds and disks busy more than a third of the time are likely to be causes of poor system performance. Excessive wait times can be the fault of the disk controller rather than the disk itself and attention should be paid to the number of devices attached to the controller in question.

Table 10.9 iostat Extended Statistics Headings

HEADING	DESCRIPTION
disk	The disk device
r/s	Read transfers per second
w/s	Write transfers per second
kr/s	Kilobytes read per second
kw/s	Kilobyte written per second
wait	Average number of commands waiting in the device queue
actv	Average number of commands in process
svc_t	Service time
%w	Percentage of time waiting in the queue
%b	Percentage of time the device is busy

The `iostat -xPnce` command that we looked at earlier to see disk activity associated with our swap partition provides additional extended diagnostics. These are described in Table 10.10.

fusage

The `fusage` command reports on the block I/O activity of local disks. In the first of the two examples shown below, the `fusage` command is used to determine what process is using the `/var/adm/messages` file. Not surprisingly, the identified process is `syslogd`. In the second example, we see that `fuser` is used to determine what process is using the executable file `dtterm`. Again, the response is no surprise. The process itself has its executable open.

```
# fuser /var/adm/messages
/var/adm/messages:      211o
# ps -ef | grep 211
    root   211     1  0 18:04:19 ?        0:00 /usr/sbin/syslogd

# fuser /usr/dt/bin/dtterm
/usr/dt/bin/dtterm:      442tm
# ps -ef | grep 442
    root   442   441  0 18:05:11 ??       0:03 /usr/dt/bin/dtterm
```

Monitoring Your Network

Network performance is an extremely important aspect of systems performance. An otherwise responsive server can be rendered unable to provide clients with adequate service if network congestion makes efficient communication impossible.

Before we look at commands that can be used to gauge the state of the local network and the server's connections, we will take a quick look at how network connections are created, how they persist, and how they are eventually shut down.

Table 10.10 Additional iostat Headings

HEADING	DESCRIPTION
s/w	Soft errors
h/w	Hard errors
trn	Transport errors
tot	Total errors
device	Device for which the statistics are being reported

Figure 10.2 illustrates the states that TCP connections pass through during the life span of a normal connection. The typical network connection will start with a SYN_SENT (a request to make a connection) and quickly move into the ESTABLISHED state in which the connection will stay while the work of the connection is occurring. This follows the famous three-way handshake in which the connection is set up. This could be any TCP connection—it might be a Telnet session or an email exchange, for example.

When the work of the session is complete, a FIN (finished) signal is sent requesting that the connection be closed. As FIN and ACK (acknowledgement) signals are exchanged, the connection moves into a FIN WAIT-1 and then a FIN WAIT-2 or CLOSING state. It then goes to a CLOSE WAIT where it remains until the connection is actually closed. When a connection is closed, all network resources that it has used are freed.

One of the important things to note is that some state transitions occur because signals are sent from client to server or vice versa. Other transitions merely depend on timing. After so many seconds, a TIME_WAIT connection is closed. The other important thing to note is that connections take system resources. Even those that no longer represent any exchange of data between the server and client occupy a file descriptor—the same resource that is used to keep track of open files. In other words, the resources used by your network connections can impact the server on a larger scale. For this reason, it is important that you know how to evaluate the connection status of your server and understand the rationale behind TCP/IP tuning. The number of possible connections and the timing of certain state transitions can be modified to improve overall performance on the server, but it can also backfire and make performance worse.

The primary command for monitoring network activity is `netstat`. We will look at three basic versions of the `netstat` command—one that provides a summary report of network activity (`netstat -s`), one which allows us to watch input and output packets and gauge the percentage of collisions (`netstat -i`), and one that examines current network connections (`netstat -a`).

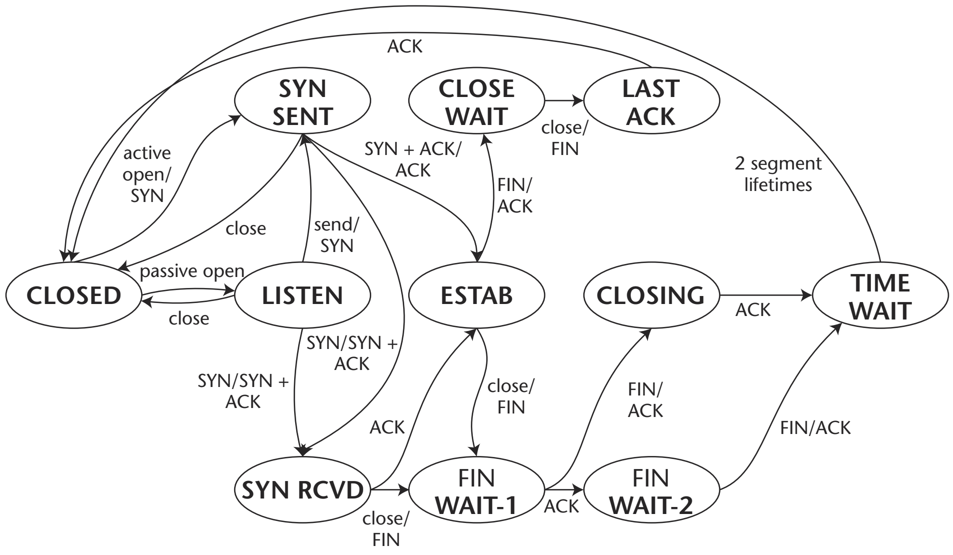


Figure 10.2 TCP state transitions.

netstat -s

The `netstat -s` command provides a summary of TCP/IP counters. Though, as with so many performance commands, the amount of data that this command provides is quite overwhelming (and you would have to be quite a TCP/IP ace to understand what all the parameters represent), there are some simple commands that you can use to highlight certain aspect of your server's network performance.

One way to extract some data for a quick analysis is to run the `netstat -s` command and pipe the output to a `grep` command to match on the string "Error." By looking at the error counts, you determine how well your interface is working. You should not see a large number of error packets.

Another item that you can extract from the volume of data is the number of packet resends. This will tell you how hard your server is having to work to send packets to other systems. Resends are only needed when there are collisions or when acknowledgements are not received after packets are sent.

netstat -i

The `netstat -i` command will give you a count of output packets and collisions. You can use these numbers to calculate the percentage of collisions. Collisions should not account for more than 5 percent of your output packets.

netstat -a

The `netstat -a` command and variants like `netstat -anP tcp` provide information on the state of current connections. A server process that is waiting for requests from clients will be in the LISTEN state. To display a list of server processes on your server that are waiting for client requests, use the `netstat -a | grep LISTEN` command.

To count up the number of connections to your Web server, use a command like this:

```
# netstat -anP tcp | grep ^192.1.2.3.80 | wc -l
```

This command will examine your network connection queue for connections on port 80 (192.1.2.3 is the IP address of the local system).

To get a quick view of your network connection status, you could write a simple script to count connections by state and report the results. The script below does this using `netstat`, `sort`, and `uniq -c` on the output of a selective `netstat` command.

```
# cat /usr/local/bin/chkNet
netstat -anP tcp -f inet | tail +5 | awk '{print $NF}' | sort | uniq -c
# /usr/local/bin/chkNet
14 ESTABLISHED
 5 IDLE
36 LISTEN
```

Alternately, you could report this kind of information through a Web page. The cgi shown here (written in Perl) summarizes network connections for port 80 (easily changed to any other port) and reports this information on a periodically updating Web page.

```
#!/usr/bin/perl

@ns=`netstat -anP tcp -f inet | tail +5`;
$est=0,$tw=0,$cw=0,$lsn=0;

print <<WEB_PAGE;
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<html>
<TITLE> Network Load </TITLE>
<h1>Connections by State</h1>
WEB_PAGE

foreach $line (@ns) {
    $_=$line;
    ($state)=/(\S+)$/;
    if ( $state =~ /ESTABLISHED/ ) { $est++;
    } elsif ( $state =~ /TIME_WAIT/ ) { $tw++;
    } elsif ( $state =~ /CLOSE_WAIT/ ) { $cw++;
    } elsif ( $state =~ /LISTEN/ ) { $lsn++;
    }
}

print "LISTEN:      $lsn <br>\n";
print "ESTABLISHED: $est <br>\n";
print "TIME_WAIT:   $tw <br>\n";
print "CLOSE_WAIT:  $cw <br>\n";
print "<P>\n";
print "</html>\n";
```

Using sar

The system activity reporter, `sar`, is a Solaris built-in that allows you to gather together a lot of performance statistics interactively or in log files for later analysis. Like numerous other performance-related commands that you're undoubtedly familiar with (e.g., `top` and `vmstat`), `sar` pulls its data from counters that are maintained in the Solaris kernel. These counters allow the system to keep track of such things as:

- CPU utilization
- Buffer usage
- Disk and tape I/O
- TTY activity
- Context switching

- System calls
- File accesses
- Process queue activity
- Interprocess communication
- Paging

By examining the value of these counters over time, we get an accurate view of how the system is performing.

Though we've referred to `sar` as a Solaris "built-in," we should clarify this with two additional points. For one, it won't necessarily be installed on every Solaris system that you encounter; the package that must be installed for `sar` to be included on your system is SUNWaccu. For the second, `sar` is not exclusively a Solaris command; you will find a similar, though not necessarily identical command on other Unix implementations—for example, Linux.

If `sar` isn't already installed on your Solaris server, you can add with the `pkgadd` command. Once you do, the set of files that will enable you to query data interactively or collect statistics over time will be available to you, but inactive. These files include `/etc/rc2.d/S21perf` (aka. `/etc/init.d/perf`), `/usr/lib/sa/sa1`, `/usr/lib/sa/sa2`, `/usr/lib/sa/sadc`, `/usr/bin/sar`, and `/var/spool/cron/crontabs/sys`. Each of these files is described in Table 10.11.

TIP **Try out `sar` on an underused system. Once you're familiar with the insights it provides and how to use its output, you can begin using it on more critical and heavily used systems.**

Table 10.11 sar Files

FILE	DESCRIPTION
`/usr/bin/sar`	Samples counters interactively or uses data collected in performance log files and reports it as specified by command-line arguments.
`/usr/lib/sadc`	Samples data and writes special markers noting when the counters were reset.
`/usr/lib/sa1`	Stores performance data in the `/var/adm/sa` file.
`/usr/lib/sa2`	Creates activity reports from `sar` data.
`/etc/rc2.d/S21perf`	Starts data collection when the system reboots.
`/var/spool/cron/crontabs/sys`	Specifies how often and when performance data will be collected.

Activating sar

`sar` is turned off by default, and the corresponding `crontab` entries are commented out. To run `sar` routinely on a system, you need to do two things. First, remove the comment (#) character on the `crontab` lines in `/var/spool/cron/crontabs/sys`. The best way to do this is with the command `crontab -e sys` (as root) or `crontab -e` (as sys). Next, uncomment the lines in the startup script `/etc/init.d/perf` (same file as `/etc/rc2.d/S21perf`) and run it.

When first installed, this file is inactive because all command lines are commented out. Uncomment the lines shown below:

```
if [ -z "$_INIT_RUN_LEVEL" ]; then
    set -- `/usr/bin/who -r`
    _INIT_RUN_LEVEL="$7"
    _INIT_RUN_NPREV="$8"
    _INIT_PREV_LEVEL="$9"
fi

if [ $_INIT_RUN_LEVEL -ge 2 -a $_INIT_RUN_LEVEL -le 4 -a \
    $_INIT_RUN_NPREV -eq 0 -a \( $_INIT_PREV_LEVEL = 1 -o \
    $_INIT_PREV_LEVEL = S \) ]; then

    /usr/bin/su sys -c "/usr/lib/sa/sadc
    /var/adm/sa/sa`date +%d`"
fi
```

Most of the lines in this script do nothing more than ascertain whether the system has just been booted. If it has, we run the `sadc` command under the username sys and create the log file for the current day. This file would be named `/var/adm/sa/sa26` if it were created on the 26th day of the month.

The other file that you need to edit is the `cron` file for the user sys. This file contains the command that will be used to sample kernel counters and store the results in a log file for later analysis. After you've removed the comment markers from the last three lines in this file, it will look like this:

```
#!/sbin/sh
#
# Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T.
#ident  "@(#)sys        1.5     92/07/14 SMI"   /* SVr4.0 1.2   */
#
# The sys crontab should be used to do performance collection. See cron
# and performance manual pages for details on startup.
#
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

As you examine these lines, you can see that the `/usr/lib/sa/sa2` script runs once a day Monday through Friday to create a daily activity report. It also runs with arguments that specify the time period its report will cover, namely 8:00 A.M. to 6:00 P.M.

(it says 6:01, but no data is collected in that last minute) at intervals of 1200 seconds (i.e., 20 minutes). This script runs at 6:05 p.m. and reports on all (because of the -A argument) of the data collected.

You will also notice that the sa1 script is run from two separate lines. One of these lines runs the script once an hour every day of the week, while the other runs the script at 20 and 40 minutes after the hour, when the hour is from 8 A.M. to 5 P.M. and only Monday through Friday. The combination of these lines cause the sa1 script to be run every 20 minutes between 8 A.M. and 5 P.M. Monday to Friday and once an hour otherwise.

When you look in the directory /var/adm/sa, you will notice that the files there follow of two naming conventions. There are sa## files and sar## files. The sa## files are data files. These files are binary files. Note that the last time these files were written to was 11 P.M. each evening. This makes sense when you consider the crontab entries. The sar## files are the reports that are produced every evening Monday through Friday. You will notice that they were all created at 6:05 P.M.

How sar Works

The data collection takes place as the user sys. The default crontab entries summarize the data every 20 minutes during the hours 8 A.M. to 5 P.M. Monday through Friday and hourly the rest of the time. Notice how this is accomplished in the first two lines. Then, at 6:05 P.M every Monday through Friday, the sa2 process runs. The default crontab entries (with comment characters removed) are as follows:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

The data activity reports can be generated via the sar command. The command sar -A will give the most detail. Here is a sample report with some of the important lines annotated for your reading pleasure:

```
SunOS c235786-a 5.7 Generic i86pc    03/14/99 <- The OS version, name of
                                                 the machine and
                                                 architecture
00:00:00    %usr    %sys    %wio    %idle     <- For each hour, the aver-
01:00:00       0       0       0       99       age percentage of time
02:00:00       0       0       0       99       that the CPU spent in
03:00:00       0       0       0       99       usr mode, state, and idle

22:00:00       1       1       1       97
23:00:01       0       0       1       99
24:00:01       0       0       1       98
Average        1       1       1       98     <- Average for the day
```

The next section shows how the disk devices on your system are running. Note that the old-style sd devices are listed.

```
00:00:00   device       %busy   avque   r+w/s  blks/s  avwait  avserv
01:00:00   cmdk0            0     0.0       0       1     0.4     6.5
           fd0              0     0.0       0       0     0.0     0.0
           sd0              1     0.0       0       6     0.0   104.7
           sd0,a            1     0.0       0       6     0.0   105.4
           sd0,b            0     0.0       0       0     0.0    41.0
           sd0,c            0     0.0       0       0     0.0     0.0
           sd0,h            0     0.0       0       0     0.0     0.0
           sd1              0     0.0       0       0     0.0     0.0
...
Average    cmdk0            0     0.0       0       2     3.1     6.6
```

This section shows the average `runqueue` and `swapqueue` sizes. The lower the number, the lower the load on the system. Also note that only the first two fields are reported by `sar`.

```
00:00:00 runq-sz %runocc swpq-sz %swpocc
01:00:00     1.0       0
02:00:00     1.3       0
03:00:00     1.0       0
04:00:00     1.0       0
...
22:00:01     1.4       0
23:00:01     1.2       0
Average      1.1       0
```

If you are running applications that utilized semaphores, that information would show up in the following section of the sar report:

```
00:00:00   msg/s  sema/s
01:00:00    0.00    0.00
02:00:00    0.00    0.00
03:00:00    0.00    0.00
04:00:00    0.00    0.00
05:00:00    0.00    0.00
06:00:00    0.00    0.00
07:00:00    0.00    0.00
08:00:00    0.00    0.00
09:00:00    0.00    0.00
10:00:01    0.00    0.00
11:00:01    0.00    0.00
12:00:00    0.00    0.00
13:00:01    0.00    0.00
14:00:00    0.00    0.00
---------
22:00:01    0.00    0.00
23:00:01    0.00    0.00
Average     0.00    0.00
```

The data provided by `sar` can be overwhelming, especially for junior system administrators still trying to get their arms around what all the numbers actually mean. For this reason, we suggest that you take a look at tools like SarCheck that evaluate the output of the `sar` command and provide human-readable reports. For some sites, this tool has not only pinpointed problems that might not have otherwise been discovered but also served as a good resource for sysadmins interested in learning about performance.

The Two Forms of sar

If you look at the man page for the `sar` command, you will note that two lines of syntax are presented. The first of these looks like the line below. This is the interactive (or *real-time*) `sar` command.

```
sar [-aAbcdgkmpqruvwy] [-o filename] t [n]
```

Though the command options listed in the first set of brackets look the same as those for the noninteractive, or "historical," version of the command, the remainder of the options are different. Interestingly, the only command argument that is not optional in the syntax shown above is the "t" (i.e., the time interval). There are no mandatory arguments for the historical version of the command. What this tells you is that, if you were to enter the command `sar` on a line by itself and hit the Enter key, you would not be invoking `sar` interactively.

The interactive command allows you to store your output in a file with the -o option. This is not the same as redirecting output. Where redirecting output would send your ASCII output to a text file, storing data with the -o option stores the information in binary form, much the same as the data that `sar` collects in data files.

The syntax for the historical `sar` command is:

```
sar [-aAbcdgkmpqruvwy] [-e time] [-f filename] [-i sec] [-s time]
```

Notice that you can supply a filename for looking at data from some earlier date. If you do not provide a filename, `sar` uses the current day's data file. You can also specify a begin (i.e., s for "start") and end time to focus in on a particular portion of a day and an interval for the time for which you want this data summarized. However, it makes little sense to use anything other than a multiple of the interval at which the data was collected. If your data was collected every 20 minutes (as is the default), look at each 20-minute interval or summarize the data every 40 minutes or every hour.

Like many other performance commands, `sar` can be run with both an interval and a count (the number of intervals). If you run the command `sar -A 10`, you will get a 10-second summary of all counters. If you use the command `sar -a 20 30`, you will get thirty 20-second summaries of file access data.

The sar Options

The various options available with `sar` are outlined in Table 10.12. More information on `sar` and an annotated `sar -A` report is included in the *How sar Works* section earlier in this chapter.

Table 10.12 sar Options

OPTION	DESCRIPTION
`-a`	File system access statistics
`-A`	All data
`-b`	Buffer activity
`-c`	System calls
`-d`	Disk activity
`-e hh:mm:ss`	End time
`-f filename`	File or data source
`-g`	Paging activity (e.g., page outs, page scans)
`-i sec`	Data summary interval
`-k`	Kernel memory allocation
`-m`	Message and semaphore activity
`-o filename`	Output file
`-p`	Paging activity (e.g., page faults, page ins)
`-q`	Run queue length
`-r`	Unused memory pages and disk blocks
`-s hh:mm:ss`	Start time
`-u`	CPU utilization
`-v`	Status of process, inode and file tables
`-w`	System swapping and switching activity
`-y`	TTY device activity

For example, in the sample output below, we've asked for five separate 5-second reports.

```
# sar -u 5 5

SunOS rollerdisco 5.8 Generic_108528-05 sun4u    09/03/02

14:32:38    %usr    %sys    %wio   %idle
14:32:43       1       1       1      98
14:32:48       0       1       0      99
14:32:53       0       0       1      99
```

```
14:32:58       0        0        1       99
14:33:03       5        1        0       94

Average        1        1        0       98
```

This interactive command takes five samples of CPU activity and displays counters for user time, system time, waitio, and idle.

Beyond sar

The `sar` command can provide a lot of insight into what your system is doing. At the same time, the data doesn't analyze itself. You can find yourself spending a lot of time looking at the data and trying to relate the numbers to specific bottlenecks on your system. To assist with this problem and the problem of data overload in general, a number of tools have been developed to simplify the process of turning data into answers. We would like to mention two of them: the SE toolkit and SarCheck.

Virtual Adrian and the SE Toolkit

The SE toolkit is an interpreted language for building performance tools and utilities. Developed by Adrian Cockcroft and Richard Pettit, the SE toolkit and Virtual Adrian attempt to bring the wisdom of a performance expert to bear on your system's data. In other words, it provides many of the rules of thumb that experts use in differentiating good from bad performance.

Of course, you can add your own tools or fiddle with the ones included with the software. The kit takes a modest amount of setup and time to become familiar with how it works. The SE toolkit is free and available from www.setoolkit.com.

SarCheck

An inexpensive tool that can help you interpret `sar` data and look for bottlenecks is SarCheck from Aptitune. SarCheck looks for memory shortages and leaks, disk load imbalances, CPU bottlenecks, runaway processes, and improperly set kernel parameters.

SarCheck can take `sar` data and convert it into a form that can be plotted with gnuplot or presented as a Web page. It can also provide detailed reports analyzing your system and make incremental recommendations for improving performance.

SarCheck is simple to install. Because this tool primarily uses data that is already collected by `sar` (along with `ps` output and kernel data), it adds virtually no overhead to your system.

System Tuning

Tuning a system for better performance requires insight into the way that the system works and the effect that your changes will have. As with any significant system change, always be prepared to back out any changes that you make in case they have a detrimental effect on performance instead of the effect that you intended.

Kernel Tuning

When your analysis of system performance indicates that your CPU is spending far too much time on system activity, it may be time to tune your kernel. Though kernel tuning is an art, there are a few parameters that you can modify without having to delve deeply into how the kernel operates.

maxusers

The simplest kernel parameter to modify and the most often tuned is maxusers. The maxusers parameter is used to size a number of kernel parameters at once. As its name implies, this parameter was initially meant to correspond to the number of simultaneous users you expected might be on your system.

maxusers is set by default to your memory size in megabytes, but not higher than 1024. In other words, if you have 4 GB of memory, maxusers would be set to 1024. If you have 512 megabytes, it would be set to 512.

The kernel parameters that are set according to the value of maxusers (unless specifically set otherwise) are listed in Table 10.13.

File Descriptors

The number of files that can be open simultaneously by each process is determined by a pair of kernel parameters that set the soft and hard limits. The *hard limit* is the limit defined by the kernel. If your server is not configured to allow an adequate number of open files and connections, processes will reach the point at which they cannot accept any additional connections or open any additional files.

Table 10.13 maxusers Parameters

PARAMETER	HOW VALUE IS SET
`max_nprocs`	The maximum number of processes is set to 10, plus 16 times the value of maxusers.
`ufs_inode`	The size of the inode cache size is set to 4 times the sum of `maxusers` + `max_nprocs` plus 320.
`ncsize`	The size of the name lookup cache is set to four times the sum of maxusers + max_nprocs plus 320.
`ndquot`	Quota table size is set to (10 times the value of `maxusers`) plus `max_nprocs`.
`maxuproc`	The user process limit is set to the value of `max_nprocs` minus 5.

The defaults for these parameters have changed considerably with each subsequent version of Solaris, as the following table shows.

RELEASE	RLIM_FS_CUR	RLIM_FS_MAX
Solaris 7	64	1,024
Solaris 8	256	1,024
Solaris 9	256	65,536

CAUTION You should not set the limits for `rlim_fs_max` arbitrarily high without looking into the repercussions for various system calls that use these limits.

The limits are set through use of the `rlim_fs_cur` and `rlim_fs_max` parameters in the `/etc/system` file. The `limit` or `ulimit` commands can then be used to set the file descriptor limits for any user or process up to the hard limit defined. To use the limit command with `csh`, you might do this:

```
% limit descriptors 1024
```

To use the ulimit command with Bourne or ksh:

```
$ ulimit -n 1024
```

Interprocess Communication

Interprocess communication is also a contributor to overall system performance. The ease with which processes are able to share data and send messages to each other can greatly influence server performance.

Interprocess communication is implemented via three separate mechanisms. These are:

- Shared memory
- Semaphores
- Message queues

If you want to list IPC parameters, you can use the `sysdef -i` command as shown here:

NOTE This is the bottom of the output. Most of the output has been omitted.

```
*
* IPC Messages
*
```

```
      2048  max message size (MSGMAX)
      4096  max bytes on queue (MSGMNB)
        50  message queue identifiers (MSGMNI)
        40  system message headers (MSGTQL)
    *
    * IPC Semaphores
    *
        10  semaphore identifiers (SEMMNI)
        60  semaphores in system (SEMMNS)
        30  undo structures in system (SEMMNU)
        25  max semaphores per id (SEMMSL)
        10  max operations per semop call (SEMOPM)
        10  max undo entries per process (SEMUME)
     32767  semaphore maximum value (SEMVMX)
     16384  adjust on exit max value (SEMAEM)
    *
    * IPC Shared Memory
    *
      33554432      max shared memory segment size (SHMMAX)
         1  min shared memory segment size (SHMMIN)
      4096  shared memory identifiers (SHMMNI)
       256  max attached shm segments per process (SHMSEG)
    *
    * Time Sharing Scheduler Tunables
    *
    60      maximum time sharing user priority (TSMAXUPRI)
    SYS     system class name (SYS_NAME)
```

To see currently allocated shared memory, semaphores, and message queues, use the `ipcs -a` command.

Shared Memory

If you think of a running process as having three parts—the code, the data, and the control information (or *heap*)—shared memory is the mechanism by which process data is shared. In fact, sharing memory is the most efficient way that processes can pass data. Sharing data not only reduces the amount of memory that is used by an application, it also greatly reduces I/O. Memory sharing is only possible when processes are running on a single system. Though shared memory may contain many segments for many different processes, a process can only write to a shared memory segment if it knows the key and has write permission and can only read from shared memory if it knows the key and has read permission. In other words, an Oracle shared memory segment will only be shared by Oracle processes.

Some applications, in particular databases like Oracle, make heavy use of shared memory. There is a standard set of lines that just about every Oracle DBA will add to the /etc/system file to ensure that the shared memory feature of Solaris is well suited to Oracle. These lines will look, more or less, like this:

```
set shmsys:shminfo_shmmax=2147483648
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
set semsys:seminfo_semmni=100
set semsys:seminfo_semmsl=100
set semsys:seminfo_semmns=1024
set semsys:seminfo_semopm=100
set semsys:seminfo_semvmx=32767
```

The first four lines relate to shared memory. These lines do *not* allocate a memory segment, but set a limit on the size of segments that can be allocated by the kernel. By adding these lines to /etc/system, you are effectively tuning the kernel, although the changes will not be effective until the system is rebooted.

NOTE The value of shminfo_shmmax should not exceed your physical memory. The settings shown above reflect a system with 4 GB of physical memory.

The shared memory parameters are described in Table 10.14.

To view shared memory, use the ipcs -mb command as shown here:

```
ipcs -mb
IPC status from <running system> as of Mon May 20 18:27:46 GMT 2002
T         ID      KEY        MODE        OWNER    GROUP      SEGSZ
Shared Memory:
m        200   0xec4e0918 --rw-r-----   oracle      dba  107143168
m          1   0x500487d7 --rw-r--r--     root     root          4
```

Table 10.14 Shared Memory Parameters

PARAMETER	DESCRIPTION
shmmax	Maximum size of a shared memory segment. No other kernel resources are influenced by this setting.
shmmin	Smallest possible shared memory segment.
shmmni	Maximum number of shared memory identifiers.
shmseg	Maximum number of segments per process. Usually set to same value as shmmni.

Semaphores

Semaphores are another form of interprocess communication and, as the name implies, are more like a signaling mechanism than a shared resource. The kernel parameters for configuring semaphores are also tuned in the `/etc/system` file and are explained in Table 10.15.

Message Queues

Message queues are similar to semaphores, but they are used for asynchronous message passing. The parameters that you can set are outlined in Table 10.16.

Solaris Resource Manager

Bundled with Solaris 9, the Solaris Resource Manager (SRM) provides a giant step beyond monitoring how system resources are being used and trying to keep services running to meet the service levels that your user community expects. SRM provides tools for allocating the resources that an application will use and better monitoring their use of those resources. It also allows better use of this information for capacity planning and billing.

Table 10.15 Semaphore Settings

PARAMETER	DESCRIPTION
`semmap`	Number of entries on the semaphore "map"—should be smaller than `semi`.
`semmni`	Maximum number of systemwide semaphore sets.
`semmns`	Maximum number of semaphores in the system (should be set to equal `semmni` times `semsl`).
`semmnu`	Maximum number of undo structures in the system—should be set to same value as `semmni`.
`semmsl`	Maximum number of semaphores per semaphore set.
`semopm`	Maximum number of semaphore operations that can be performed in each `semop` call.
`semume`	Maximum number of undo structures per process.
`semusz`	Number of bytes required for `semume` undo structures.
`semvmx`	Maximum value of a semaphore.
`semaem`	Maximum adjust-on-exit value.

Table 10.16 Message Queue Settings

PARAMETER	DESCRIPTION
`msgmap`	Number of entries in the msg map. The default is 100; the maximum is 2 GB.
`msgmax`	Maximum size of a message. The default is 2048; the maximum is 2 GB.
`msgmnb`	Maximum number of bytes for the message queue. The default is 4096; the maximum is 2 GB.
`msgmni`	Number of unique message queue identifiers. The default is 50; the maximum 2 GB.
`msgssz`	Message segment size. The default is 8; the maximum 2 GB.
`msgtql`	Number of message headers; The default 40; the maximum 2 GB.
`msgseg`	Number of message segments. The default 1024; the maximum is 32 KB.

What Is SRM?

Combining portions of the earlier SRM and the Solaris Bandwidth Manager along with some significant additional capabilities, the SRM in Solaris 9 is anything but optional. Instead, it is so well integrated into the operating system that it affects a user from the time that he or she logs in—in fact, whether or not the user gets logged in.

Solaris Resource Manager allows a system administrator to deny, limit, or reserve resources so that critical processes are never deprived of resources that they require and misbehaving processes are limited in the damage that they can cause. It works by establishing a series of constraints, scheduling times, and partitioning under which processes are controlled.

Constraints. Setting bounds on resource consumption.

Scheduling. Controlling the allocation of resources when demand is high.

Partitioning. Reserving a portion of resources for particular services.

With Solaris Resource Manager, sysadmins are able to provide different levels of service to different applications. They can, for example, isolate Web sites from each other's traffic. Differing service levels can be provided on a single box.

With SRM, sites with large servers have additional options for managing servers beyond moving busy application to less busy boxes or using separate resources to balance demand. They can now provide adequate (i.e., policy-driven) service to numerous applications on a single box.

How Does SRM Work?

We can only provide an introduction to the Solaris Resource Manager in this book. Full coverage would require several chapters if not an entire book. We will, however, outline the service, explain the commands and files that compose the service, and explain how some system commands have been modified to accommodate the concept of projects.

The primary file used in managing projects is `/etc/project` file. This colon-delimited file contains the following fields:

```
projname:projid:comment:user-list:group-list:attributes
```

where:

projname is a string that names the project.

projid is the project's numeric ID.

comment is your description for the project.

user-list is a comma-delimited list of the users associated with the project.

group-list is a comma-delimited list of the groups associated with the project.

attributes is a series of resource limits that are applied to the project.

The `/etc/project` file can be provided via NIS or LDAP. It looks like this when you first install a system:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

When a user logs in, his or her login session is associated with a project. If a project cannot be associated with the user, the login is unsuccessful. The process of connecting a user to a project proceeds as follows:

First, the system looks in the `/etc/user_attr` file to see if the user has an entry. If so, the value of the project attribute from that file is used. The `/etc/user_attr` file looks more or less like this:

```
#
#pragma ident       "@(#)user_attr 1.2     99/07/14 SMI"
#
root::::type=normal;auths=solaris.*,solaris.grant;profiles=All
shs::::profiles=System Administrator,Primary
Administrator;roles=whatever;type=normal
```

Failing that, the system looks in the `/etc/project` file for an entry of the form `user.username`. If it finds such an entry, the specified project is used.

Failing that, the system looks in the `/etc/project` file for an entry of the form `group.groupname`, where the group matches the user's default group. If it finds such an entry, the specified project is used.

Failing that, the system looks for a default project (e.g., `default:3::::`) in the `/etc/project` file. If no default entry exists, the login fails.

Project-Management Commands

A number of new commands and new options for old commands provide the tools for managing projects. Each of these is briefly introduced below. We encourage the reader to refer to the man pages for these commands for additional information.

projects

The `projects` command lists the projects that a user is associated with. An example of such a command is shown here. We are requesting a list of projects for the user shs.

```
# su - shs
# projects
perfmgt
```

newtask

The `newtask` command executes a user's default shell or specified command, creating a new task within the specified project.

```
# newtask -v -p perfmgt
    19
```

The `-v` option means verbose and causes the command to print the task ID associated with the new task.

To add a running process to a project, you would also use the `newtask` command, but you would add the `-c` option to identify the process you are adding as shown in this example:

```
# newtask -v -p perfmgt -c 1234
    21
```

The response, 21, identifies the task ID associated with the process that was just added.

projadd

The `projadd` command adds a new project to the `/etc/projects` file. For example, the following command adds a project named perfmgt, assigns it project ID 1112, and assigns the project to the user shs:

```
# projadd -U shs -p 1112 perfmgt
```

projmod

The `projmod` command modifies the project's information on the local system. In this example, we are adding a description for the project:

```
# projmod -c 'Performance Mgt' perfmgt
```

projdel

The `projdel` command deletes a project from the local system. In this example, we remove the project we created and modified in the earlier examples:

```
# projdel perfmgt
```

pgrep

The `prgep` command has been augmented to provide process information for projects. The following command lists the process ID associated with the `perfmgt` project:

```
# pgrep perfmgt
    1234
```

The following commands are also available:

```
pgrep -J <project-Id>
pgrep -T <task-Id>
```

pkill

The `pkill` command, used for killing a process without having to first determine its process ID, can also be used with projects as shown in the following commands:

```
pkill -J <project-Id>
pkill -T <task-Id>
```

prstat

The `prstat` command which, as we described earlier in this chapter, provides information on processes also provides information on projects when used with the `-J` option. This information will appear below the information provided for processes.

```
# prstat -J
```

prctl

The `prctl` command allows you to display and modify resource controls with a local scope. For example, to display the maximum number of file descriptors that the current shell may use, issue a command like this:

```
# prctl -n process.max-file-descriptor $$
```

NOTE Notice that `$$` always represents the current process.

rctladm

The `rctladm` command allows you to list or modify system behavior associated with resource limits with a global scope. For example, the command will provide information about what the system will do when the threshold for use of the specified resource is exceeded.

```
# rctladm process.max-cpu-time
```

If `no-deny` is the response, you know that access to the CPU is not denied.

su

To switch to a user's default project, use the `su - username` command. An `su` command without the dash will retain the current project assignment.

id

The `id -p` command prints the project identifier along with the UID and GID. For example, the following command provides information for the current project:

```
# id -p
uid=111(shs) gid=14(sysadmin) projid=1112(perfmgt)
```

ps

The `ps -o` command displays tasks and projects, as follows:

```
# ps -o
USER PID  UID  PROJID
shs  1234 111  1112
# ps -o project,taskid
PROJECT TASKID
perfmgt 1112
```

Table 10.17 Resource Limits

CONTROL	RESOURCE
`project.cpu-shares`	Number of CPU shares granted
`task.max-cpu-time`	Max CPU time
`task.max-lwps`	Max LWPs
`process.mac-cpu-time`	Max CPU time
`process.max-file-descriptor`	Max file descriptors
`process.max-file-size`	Max file offset
`process.max-core-size`	Max size core file process can create
`process.max-data-size`	Max heap memory
`process.max-stack-size`	Max stack memory segment
`process.max-address-space`	Max amount of address space

Resource Limits

Table 10.17 lists the resources that are constrained by the listed controls.

Implementing Resource Control

We have presented an extremely brief overview of the SRM because of space limitations. Decisions on how to allocate resources on your systems must be based on knowledge of the services you support and the resources at your disposal. We recommend that you consult more detailed information on SRM before attempting to implement it on your production servers.

Summary

This chapter covered a lot of territory. We began with the reasons why performance analysis is one of the more difficult aspects of systems administration. We explained bottlenecks. We introduced numerous commands for obtaining information on system performance along with some hints on how to interpret the data. We gave you a lot of information on a lot of commands and some shortcuts that will help you to more quickly troubleshoot performance on your systems.

We introduced Solaris Resource Manager, along with the concepts of projects and tasks. The new project orientation of Solaris will have a huge impact on the way that busy servers are managed. We hope that we have helped you get started in the right direction.

CHAPTER
11

Volume Management

One of the primary responsibilities of Unix system administrators in the decades since Unix systems first came into being has been managing file systems—creating them, backing them up, mounting them, sharing them, and looking for ways to clean up space so that they would not fill up to capacity and become unusable. Users still don't do a good job of removing files they no longer need and our applications are increasingly data hungry. Even so, many system administrators are finding that the job of managing file systems has taken a dramatic turn for the better. Instead of, or maybe only in addition to, looking for core files and other likely candidates for removal so that space can be freed up, they are using tools that allow them to add disk space on the fly. Instead of, or maybe in addition to, backing up files to tape every night, they are preventing loss by storing files with some level of redundancy. In short, they are using volume management.

Volume managers free system administrators from the confines of spindles and partitions and allow them to manage their disk space in response to demand. A file system far larger than any single disk on a server can be created from a series of small disks. File systems can be grown as needed and without shutting a server down or interrupting ongoing activity. Reliability and performance can be improved by using the right RAID level for the job. Hot spares can be ready and waiting for the next data crunch.

This chapter introduces the concepts and terminology used in volume management, provides just enough information on the key volume management tools for Solaris—Veritas Volume Manager and Sun's Solaris Volume Manager—to make you want to jump in to the fray, and offers some insights and suggestions for making the best use of this technology.

Volume Management Concepts and Terminology

A volume manager can be said to be converting I/O requests into I/O requests for the underlying devices. Some terms that we use in describing volume managers, therefore, deal with the underlying hardware, while others deal with the virtual layer through which I/O requests are funneled. Figure 11.1 illustrates this concept.

Following is a list of volume management terminology:

Volume. One of the most important terms used in volume management is, of course, the term *volume*. A volume is a virtual disk. In other words, it is the unit of storage in a volume management system that is used as if it were a physical disk. Sometimes referred to as a *metadevice*, a volume not only *behaves* as a disk, but is used as if it *were* a disk by applications and by file systems. Below the surface, a volume is a group of "plexes" (a term defined later in this list). These plexes may be striped or mirrored.

Volume manager. A tool or set of tools that enable you to create and manage volumes created from your physical disks. Volume managers can be thought of as introducing a layer between the files systems and the physical disks.

Slices and **partitions.** These terms both refer to portions of physical disks, but are subsumed into volumes. Other terms that relate to the physical disks are sectors, blocks, tracks, cylinders, and LUNs (logical unit numbers). When these terms are used, you know people are talking about the physical disks and not volume management.

Subdisk. A portion of a disk that is managed by the volume manager. Unlike a partition, a subdisk does not, therefore, relate to the underlying physical partitions on the disk.

Soft partition. A subdivision of a partition, generally used to provide smaller, more manageable storage units for certain operations.

Transactional volume. Used to provide logging for UFS file systems when logging is enabled.

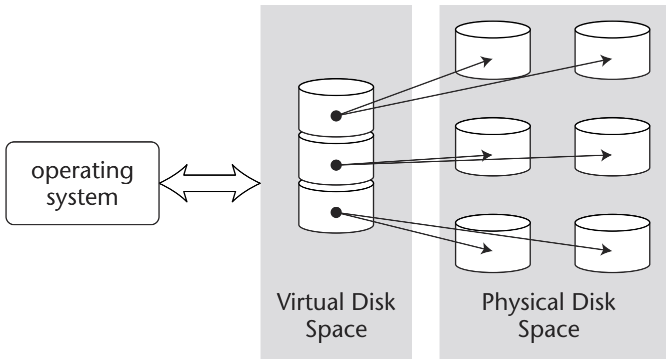


Figure 11.1 Virtual disk space.

Concatenation. The process of combining subdisks in an end-to-end arrangement. Space in a concatenation is used sequentially, such that the first subdisk is used before data is stored on the next and so forth. Concatenation is one way of forming a large virtual disk from a number of smaller disks and works well for small random I/O.

Stripe. A logical storage space comprising a portion (or “strip”) of a number of physical disks. A *stripe set* is the array of stripes that form a striped RAID volume. Striping is generally used because it speeds up disk I/O. Data from stripes is stored and retrieved more uniformly than is the case with concatenation. In other words, data may be retrieved from a number of stripes simultaneously, resulting in a decrease in how much time the I/O operation takes to complete.

Interlace. A value that specifies, in essence, how wide a stripe is—that is, how much data it writes to each component before going on to the next.

Mirror. A disk or portion of a disk that is an exact replica of another. In mirrored setups, you have two copies of every file, though this is not obvious from looking at the file systems. All write operations affect both members of the mirrored set. If one or the other of the disks or partitions in the mirrored set fails, a volume manager will revert to using the other. If you inadvertently remove a file that you need, mirroring does *not* help because both copies will have been removed. Either disk can service a request, so there is some performance boost as well. A *submirror* is one component of a mirror.

Parity. A computationally inexpensive way to correct single errors in data. The value that results from the XOR (one and not the other) bit-wise operation of a series of numbers can be similarly combined with *N*-1 of these numbers (where *N* is the number of items in the set) to compute the missing number. Parity is, thus, used in certain RAID levels to make it possible to reconstruct data that is corrupted or lost when a disk is damaged. For example, if data is striped across a set of four disks and a fourth contains parity, the system can withstand loss of any of the four disks because the parity data can be used to reconstruct the missing data.

Plex. A group of one or more subdisks, typically from different physical drives, that are used together in a concatenation or striping arrangement.

Raw partitions. Disk partitions that house data but do not contain file systems. They are often used by applications that perform better without the overhead of file system data structure such as inodes and directory structures.

State database. A storage area in which a volume manager maintains information about the virtual disks or volumes. For reliability, state database replicas are duplicated across the set of partitions composing the volume. Another term used for a state database is *private region*.

Hot spare. A drive that is installed and ready to be substituted for an active disk that fails. Adding a hot spare to the set of disks that are in use (for example, to take the place of one that has failed) does not require a shutdown of the server. A *hot spare pool* is a group of such drives that can be used as needed.

Resynchronization. What a volume manager does when it must rebuild a failed drive on a hot spare.

Encapsulation. What Veritas Volume Manager does to a preexisting partition to turn it into a Veritas volume. It preserves the data on the disk, while grabbing one or more cylinders for the private region.

Array. A logical disk comprising of a number of physical disks, typically housed in a rack-mountable enclosure. A number of disks may be designated as hot spares.

RAID. RAID stands for *redundant array of inexpensive disks*. RAID systems provide improved performance, protection against disk failure, and space for very large file systems. All RAID levels except for RAID 0 (striping) provide a way to reconstruct data when a drive fails.

Chunk. A RAID term that refers to the smallest retrievable amount of storage from a RAID volume. Various RAID types are described in Table 11.1.

Degraded mode. This term means that a RAID device is running at reduced performance due to a failure. For example, if one drive in a stripe with parity setup has failed, every operation may involve reconstructing the missing data.

Hardware RAID. A setup in which the physical disks are invisible to the system. All aspects of volume management are handled by the RAID device.

Disk controller. The hardware that issues commands for reading from and writing to the disk. Disk controllers often contain a cache, which is fast-access memory that serves as an intermediary data store to speed up I/O.

Random access I/O. Any I/O in which each read or write operation is independent of the others. Examples include database operations and general file server accesses.

Table 11.1 RAID Levels

RAID LEVEL	DESCRIPTION
0	Simple striping or concatenation (or both)
1	Mirroring
0+1	Striping then mirroring, also referred to as *01* and *stripe-mirror*
1+0	Mirroring then striping, also referred to as *10* and *mirror-stripe*
2,3,4	Striping with parity (very few storage environments support these)
5	Striping with parity, where parity is spread across the disks

Sequential Access I/O. Any I/O in which sequential blocks of data are likely to be read. Examples include operations in which large files are processed.

Load balancing. Refers to any technology that distributes demand for resources among a series of same-type components. With respect to volume management, load balancing might refer to the ability of a volume manager to balance access to a volume through multiple controllers.

Multipath device. Any device that can be reached through more than one same-type hardware component, such as a disk controller.

Solaris Volume Manager

Bundled into Solaris 9, the Solaris Volume Manager gives Solaris system administrators a set of tools for managing disks and file systems with a greater flexibility than ever before. Based on Solstice DiskSuite, a product that Sun has been offering for roughly a decade, the tool is familiar to many system administrators and in its current incarnation is free.

Solaris Volume Manager provides three basic advantages over conventional disk partitioning and file system creation. First, it increases data availability by creating volumes with some degree of internal redundancy. Second, it improves disk performance when certain options (e.g., striping) are used. Third, it allows system administrators to respond to disk space shortages without system downtime.

Sun likes to point out that Solaris Volume Manager is *integrated* into the operating system. This is not just one of those buzzwords that marketers like to use. Integration means two things that may be of considerable value to you. One of these is that, when using Solaris Volume Manager, you will deal with a single vendor. You will never have the volume manager vendor blaming the operating system for a problem that you encounter, and vice versa. The other is that you can expect a tight coupling between the volume manager and the operating system. They will have been developed by people who talk to each other early in the development process and on a continuing basis. If the operating system takes a right turn somewhere deep inside the kernel, the volume manager is likely to remain on the right track. Even with Sun's dedication to open standards, this coupling just might equate to better performance and reliability. Whether this turns out to be the case is something that none of us can predict.

Features

Solaris Volume Manager supports both striping and concatenated volumes. These technologies help to distribute the demand for data over a number of disks and possibly multiple controllers, speeding up data access. Solaris Volume Manager also supports mirroring and striping in the RAID 0+1 or RAID 1+0 configurations. RAID 1+0 is used by default if the underlying hardware is able to support it.

Soft Partitioning

Using soft partitioning, as many as 8,192 partitions can be established on a single volume. These can be established on a physical disk or on a RAID volume. The benefit to using software partitions is that they can be created and resized as needed. They also provide the administrator with a way to establish almost any number of file systems with different characteristics. For example, some may be mounted as read-only, while others are mounted with UFS logging enabled. Need a number of small partitions so that you can protect applications from other applications' files? Consider using soft partitions.

Live Volume Expansion

Solaris Volume Manager allows a volume to be expanded without downtime provided that the additional space is available. This is often referred to as *growing* a volume. Using the graphical interface or command-line counterparts, a system administrator can also create volumes that span multiple disks. Instead of replacing a series of small drives with a large drive, you can combine the drives into a single volume and get the same space with potentially better performance.

Device IDs

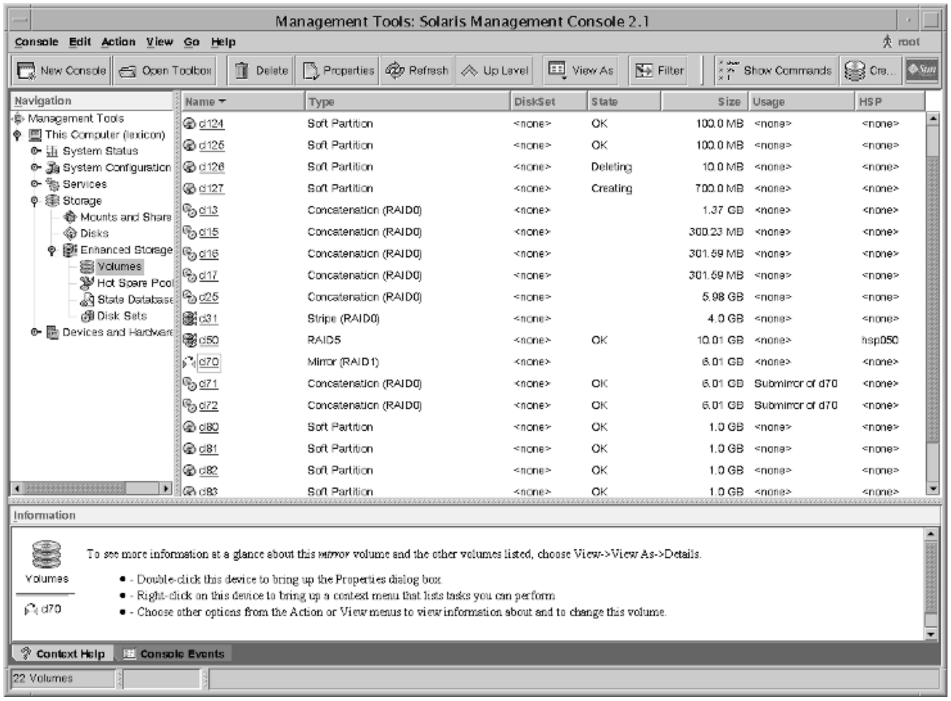
Solaris Volume Manager provides a naming mechanism that allows disks to be addressed by device IDs, simplifying configuration changes. Device identifiers for Solaris Volume Manager volumes have names such as "d11" and "d30" instead of the "c0t0d0s3" naming convention used by physical drives.

Disk Sets

Solaris Volume Manager allows you to set up disk sets. A *disk set* is a set of disk drives containing volumes and hot spares that can be alternately shared between a number of hosts as needed.

Solaris Management Console

The Solaris Management Console provides access to volume configuration through its Enhanced Storage Tool. This tool is displayed in Figure 11.2. Alternately, commands can be entered on the command line to effect the same configuration and controls. Solaris Volume Manager commands are listed in Table 11.2. The only caution you need to keep in mind is that both the GUI and the command-line tools should not be used at the same time.



Name	Type	DiskSet	State	Size	Usage	HSP
d124	Soft Partition	<none>	OK	100.0 MB	<none>	<none>
d125	Soft Partition	<none>	OK	100.0 MB	<none>	<none>
d126	Soft Partition	<none>	Deleting	10.0 MB	<none>	<none>
d127	Soft Partition	<none>	Creating	700.0 MB	<none>	<none>
d13	Concatenation (RAID0)	<none>		1.37 GB	<none>	<none>
d15	Concatenation (RAID0)	<none>		300.23 MB	<none>	<none>
d16	Concatenation (RAID0)	<none>		301.59 MB	<none>	<none>
d17	Concatenation (RAID0)	<none>		301.59 MB	<none>	<none>
d25	Concatenation (RAID0)	<none>		5.98 GB	<none>	<none>
d31	Stripe (RAID0)	<none>		4.0 GB	<none>	<none>
d50	RAID5	<none>	OK	10.01 GB	<none>	hsp050
d70	Mirror (RAID1)	<none>		6.01 GB	<none>	<none>
d71	Concatenation (RAID0)	<none>	OK	6.01 GB	Submirror of d70	<none>
d72	Concatenation (RAID0)	<none>	OK	6.01 GB	Submirror of d70	<none>
d80	Soft Partition	<none>	OK	1.0 GB	<none>	<none>
d81	Soft Partition	<none>	OK	1.0 GB	<none>	<none>
d82	Soft Partition	<none>	OK	1.0 GB	<none>	<none>
d83	Soft Partition	<none>	OK	1.0 GB	<none>	<none>

Information

Volumes

d70

To see more information at a glance about this *mirror* volume and the other volumes listed, choose View->View As->Details.

- - Double-click this device to bring up the Properties dialog box.
- - Right-click on this device to bring up a context menu that lists tasks you can perform
- - Choose other options from the Action or View menus to view information about and to change this volume.

22 Volumes

Figure 11.2 Enhanced Storage Tool (SVM GUI).

Table 11.2 Solaris Volume Manager Commands

COMMAND	FUNCTION
`growfs`	Increases the size of a UFS file system on the fly
`metaclear`	Deletes active volumes or hot spare pools
`metadb`	Creates and deletes configuration files (state database replicas)
`metadetach`	Detaches a volume from a mirror or a logging device from a transactional volume
`metadevadm`	Checks the ID configuration of devices
`metahs`	Manages hot spares and hot spare pools

(continues)

Table 11.2 Solaris Volume Manager Commands *(Continued)*

COMMAND	FUNCTION
`metainit`	Configures volumes
`metaoffline`	Puts submirrors offline (such that they cannot be used)
`metaonline`	Puts submirrors back online
`metaparam`	Modifies volume parameters
`metarecover`	Recovers configuration information for soft partitions
`metarename`	Renames volumes
`metareplace`	Replaces components in RAID 5 volumes or submirrors
`metaset`	Administers disk sets
`metastat`	Used to view volume configuration
`metasync`	Resynchronizes volumes during a reboot
`metattach`	Attaches a RAID 0 or RAID 1 component or attaches a log device to a transactional volume
`metaroot`	Sets up for mirroring root partition

Storage Monitoring

Solaris Volume Manager includes a tool for monitoring volumes. The monitoring daemon, `/usr/sbin/mdmonitord`, provides information on failed components and device errors. SNMP traps can provide information to network management tools.

`mdmonitord` is started at boot time through the `/etc/rc2.d/S95svm.sync` script. You can specify an interval (in seconds) for repeated checking to be performed by adjusting the following line as shown (causes the checking to be done every 15 minutes).

```
before: $MDMONITORD
after:  $MDMONITORD -t 900
```

Setting up Volumes

Once set up, volumes are used as if they were simply normal file systems. The only caution is that you should not use both the GUI and the command line at the same time or the result will be unpredictable. Almost any Unix command works with volumes as easily as with file systems and requires no consideration for differences in the underlying drives, except for the `format` command, which will look at the drives independently of the virtual layer provided by the volume manager.

Solaris Volume Manager can be managed via the command line (with commands such as `metainit` and `metastat`) or through the GUI provided with this tool. The GUI is available through the Solaris Management Console (SMC). Once SMC is open, from the Navigation pane at left choose This Computer→Storage→Enhanced Storage. Log in if prompted.

NOTE Only root can use the command-line interface, but users can be given privilege to use the GUI interface through the User Profile features of Solaris.

There are four types of things that you can create with the Solaris Volume Manager, including: volumes, disk sets, state database replicas, and hot spare pools.

Solaris Volume Manager requires that the state database and replicas be set up before it can do anything else. These structures contain information about the state of the volumes, including failures and configuration changes and are, therefore, critical to the proper functioning of the volume management software. Table 11.3 lists files used to manage configuration data.

What you need to know to administer Solaris Volume Manager includes:

- How to set it up
- How to view and evaluate your setup
- How to change or grow a volume
- How to know when a volume is working in degraded mode
- How to check your hot spares, including how many you have and how big they are

Drives in a single volume can be on separate controllers for generally better performance.

Table 11.3 Solaris Volume Manager Files

FILE	USE
`/etc/lvm/md.cf`	Configuration information file, updated automatically by SVM.
`/etc/lvm/mddb.cf`	Records the location of the state database replicas.
`/kernel/drv/md.conf`	Edit this file to change number of possible volumes or disk sets.
`/etc/rc2.d/S95svm.sync`	Set the error-checking interval for mdmonitord (SVM monitor daemon).
`/usr/sbin/mdmonitord`	Daemon for Solaris Volume Manager monitoring and reporting.

Set Up State Database Replicas

You must have a state database replica before you do anything else with Solaris Volume Manager. In general, it is best to have at least three—to guard against loss of this vital information—and it's best if they are on different disks or controllers. Some people suggest that you keep two replicas on every disk. You can create database replicas by using the `metadb` command or with the GUI tool (if logged in as root or enabled through user profiles).

State database replicas contain information about volumes, hot spares, and disk sets. If all replicas were lost, you could theoretically lose all the data housed on your managed volumes. The volume manager will try to recreate replicas that it deems are corrupted. If replicas disagree, for example, there is an algorithm, called the majority consensus algorithm, which it uses to determine which replica is corrupt.

In general, all replicas are updated simultaneously. Each takes 4 MB (8,192 blocks of 512 bytes) of disk space by default. The system will panic if fewer than half of the replicas are valid on a boot and will not boot unless a *majority* of the replicas are valid. This is the reason that people suggest that you have three or more. With only two replicas, a single replica failure could keep your system from booting.

You can put state database replicas on slices dedicated for this use or on slices that will be incorporated into a volume (SVM will know they're there and will not overwrite them). A slice can contain more than one replica, but your strategy should be to set up your replicas such that you are unlikely to lose all of them. Separating replicas across disks and controllers lessens the chance that a hardware failure will make your volumes unusable.

You can create database replicas with the Enhanced Storage Tool by choosing Action→Create Replicas or with the `metadb` command in this form:

```
metadb -a [-c #] [-l ####] -f slice
```

where:

`-a` specifies this is an add operation.

`-c #` indicates the number of replicas to be added.

`-l ####` indicates the size of the replicas in blocks.

`-f` forces the operation (causes it to run even if no replicas yet exist).

`slice` is the name of the component that will contain the replica.

For example:

```
# metadb -a -f c0t1d0s3
```

You can also use `metadb` to display information about the replicas you set up. To check on the status of your replicas, use the `metadb -i` command. This will list replicas, their locations and status, and will also provide a key so that you can interpret what each of the status indicators means. For example, you might see a line like this:

```
a m  p  luo    16      8192     /dev/dsk/c0t1d0s4
```

This output tells you that the replica is active (`a`), has been selected for input (`m`), and has been patched in the kernel (`p`). The value `16` is the block address, and `8192` is the size in blocks. `/dev/dsk/c0t1d0s4` is the partition where this replica is stored.

The possible keys are listed in Table 11.4.

Replicas can be deleted with the `metadb -d` command (e.g., `metadb -d -f c0t1d0s4`).

Identify Slices and RAID Level

Before you establish volumes, you need to identify the slices that you are going to put under Solaris Volume Manage control and determine the RAID level that you want to use. You can then create volumes, disk sets, additional replicas, and hot spare pools.

NOTE If you are working with slices that are already in use, you should first back up any data that they contain.

Table 11.4 metadb Status Indicators

STATUS	MEANING
r	Does not have relocation information
o	Active prior to last database replica configuration change
u	Up to date
l	Locator was read successfully
c	Location was in /etc/lvm/mddb.cf
p	Patched in the kernel
m	Master
W	Write errors
a	Active
M	Problem with master blocks
D	Problem with data blocks
F	Format problems
S	Too small to hold current database
R	Read errors

SVM allows you to create volumes of any of the following types:

RAID 0. Data is spread across disks in small, equally sized chunks, providing a high data transfer rate but no resistance to data loss.

RAID 5. Data is duplicated across multiple drives, requiring twice as much disk space, but providing considerable resistance to data loss.

Transactional. Used to log UFS file systems, these contain a master and a logging device.

Soft partition. A division of a slice or volume into a number of smaller extensible volumes.

RAID 0+1 and 1+0. Data is both striped and mirrored, providing both high transfer and resistance to data loss. These are set up from RAID 0 or RAID 1 volumes. The difference between 1+0 and 0+1 will be discussed later in this chapter.

Using RAID 0

RAID 0 provides for stripes and concatenations. If you select components on different drives and controllers, you can expect better performance.

WARNING When you set up striping, you destroy the existing file system. You must back up and restore the data or use concatenation instead to preserve it.

For striping, components should be the same size or disk space will be wasted. If you understand the nature of your I/O, you can set the interlacing of the stripe to match the nature of your data.

RAID 0 implies no redundancy, so you will still be as dependent on backups as before.

Concatenation works as well for I/O, which is random and in small pieces, while striping is more advantageous for data that is accessed sequentially.

NOTE RAID 0 volumes can still be mirrored, resulting in a RAID 0+1 volume.

How to Create a Stripe

If you are using the Enhanced Storage tool, choose Action→Create Volume. From the command line, use a command of this form:

```
metainit volume #-stripes components-per-stripe components... -I
interlace
```

For example:

```
# metainit d10 1 3 c0t1d0s3 c0t2d0s3 c0t3d0s3
```

Here, we are selecting the same partition on three separate disks and creating from them a striped volume. Notice that we are specifying 1 stripe and 3 components.

How to Create a Concatenation

If you are using the Enhanced Storage tool, choose Action→Create Volume. On the command line, use a command of this form:

```
metainit volume #-stripes components-per-stripe components... -I
interlace
```

For example:

```
# metainit d1 3 1 c0t1d0s3 c0t2d0s3 c0t3d0s3
```

Here, we are selecting the same three partitions, but we are forging them into a single component.

If one of the components has a file system that you want to preserve, it should be the first in the list, and you should unmount it before you issue the `metainit` command.

Expand a RAID 0 Volume

To increase the size of a RAID 0 volume, use the `metattach` command, as follows:

```
metattach volume-name components...
```

You can specify one or a number of components at the same time with a command such as this:

```
# metattach d33 c1t1d0s3 c1t1d0s4 c1t1d0s5
```

Remove a Volume

To remove a volume, issue both of these commands, specifying the volume name.

```
umount volume-name
metaclear volume-name
```

Using RAID 1

RAID 1 (mirroring) maintains duplicate copies of data. Though there are no implications for performance, RAID 1 gives your system the ability to continue running when a disk has failed. In fact, reduction to one disk of the mirrored set should barely affect performance. Volumes that are mirrors of each other are referred to as *submirrors*.

Mirroring is not restricted to two drives or partitions. You can do a three-way mirror, but most people only do two since this generally seems sufficiently secure, and additional disk space (beyond the doubling) might be hard to obtain.

Ideally, the disks you mirror will be the same size and type of disk. Note that, whenever you are working with mirrored disks, you only mount the mirror, never the submirror. In other words, you mount the virtual device, not the actual partition.

You can specify any of three *policies* for how mirrors are used:

Round robin. This is the default; this method tries to balance the demand on the submirrors.

Geometric. Divides reads by disk block address.

First. Sends all read requests to the first submirror in the set.

Similarly, writes can be done in parallel or in a series. Parallel is the default.

Before you create a mirror, create the volumes that will be mirrored.

NOTE Booting single user mode using a volume that is mirrored might cause the mirrors to appear as though they need maintenance, but this situation will correct itself when the `metasync -r` command is run during the boot.

To create a mirror, use a command of the following form to set up each of the volumes:

```
metainit volume-name #-stripes components-per-stripe components... -I
interlace
```

Next, create the mirror with a `metainit` command of the form:

```
metainit mirror-name [-m] first-mirror-name
```

where *-m* specifies that you are creating a mirror and *submirror-name* is the first component of the mirror.

Last, use a `metattach` command of this form to attach the second submirror:

```
metattach mirror-name second-mirror-name
```

Basically, to set up a mirror, you're going to do four things. The following example illustrates each step in setting up a two-way mirror between two drives.

```
# metainit d21 1 1 c0t1d0s2
d21: Concat/Stripe is setup
# metainit d22 1 1 c1t1d0s2
d22: Concat/Stripe is setup
# metainit d20 -m d21
d20: Mirror is setup
# metattach d20 d22
d20: Submirror d22 is attached
```

Creating a Mirror Using an Existing File System

If you are creating a mirror from an existing file system, you first need to create a RAID 0 volume on that partition with a command of the form:

```
metainit volume-name -f 1 1 slice
```

For example:

```
# metainit d11 -f 1 1 c0t1d0s0
```

Next, create a RAID 0 volume on a similar slice.

```
# metainit d12 1 1 c1t1d0s0
```

Then, create a one-way mirror using a `metainit` command of this form:

```
# metainit d10 -m d11
```

For any partition other than root, edit the `/etc/vfstab` file and set the mount to refer to the mirror. The entry will look something like this:

```
/dev/md/dsk/d10 /dev/md/rdsk/d10 /var ufs 2 - yes
```

Then, use the `metattach` command to connect the second submirror to the mirror.

```
# metattach d10 d12
```

Creating a Mirror Using a Root Partition

To create a mirror from the root partition, use the `metaroot` command in lieu of editing the `vfstab` file. The process will look like this:

```
# metainit -f d11 1 1 c0t1d0s0
d11: Concat/stripe is setup
# metainit d12 1 1 c0t2d0s0
d12: Concat/Stripe is setup
# metainit d10 -n d11
d10: Mirror is setup
# metaroot d10
# lockfs -fa
# reboot
```

Following the reboot, you can complete the process:

```
# metattach d10 d12
d10: Submirror d12 Is attached
```

If you want to be able to boot off the second root submirror, you should add a device alias to do so. First, acquire a long listing of the alternate boot device's path in the file system. For example:

```
ls -l /dev/rdsk/c0t2d0s0
lrwxrwxrwx 1  root root  55 Jul 6 21:53  /dev/rdsk/c1t2d0s0 ->
\../../devices/sbus@1,f8000000/esp@1,200000/sd@2,0:a
```

Then, at the at ok prompt, define and save the new alias with commands like the following:

```
nvalias backup /sbus@1,f8000000/esp@1,200000/sd@2,0:a
setenv boot-device disk backup net
nvstore
```

Booting off the mirror can then be accomplished with this command:

```
ok boot backup
```

Removing a Mirror

SVM provides for "unmirroring" but only for volumes that can be unmounted while the system continues to run. This is done with the `umount`, `metadetach`, `metaclear`, and `mount` commands, as shown in this example:

```
# umount /var
# metadetach d21 d22
d21: submirror d22 is detached
# metaclear -r d21
d21: Mirror is cleared
d22: Concat/Stripe is cleared
vi /etc/vfstab    <-- Put back the native partition name for the
unmirrored partition.
# mount /var
```

Detaching severs the association of a drive with a mirror. Putting it offline merely suspends read and write operations until it's brought back online.

A submirror can be in any of the states described in Table 11.5.

Table 11.5 Submirror States

CONDITION	DESCRIPTION
`okay`	Functioning properly
`resyncing`	Being synchronized following correction of an error
`maintenance`	I/O or open error has occurred, reads and writes disabled
`last erred`	Component has incurred error, and replication is not available

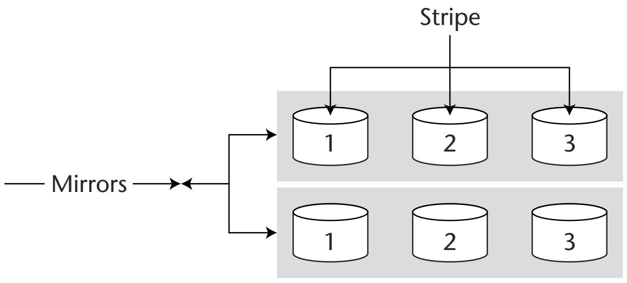


Figure 11.3 RAID 0+1.

The `metastat` command (e.g., `metastat -d11`) will allow you to see this information. For example:

```
# metastat d20
d20: Mirror
    Submirror 0: d20
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 6875765 bytes
```

RAID 0+1 versus RAID 1+0

Though RAID 0+1 and RAID 1+0 both combine the elements of striping and mirroring, there are some differences in the way these technologies are combined that make one of these RAID levels more impervious to disk failure than the other.

In RAID 0+1, also known as *stripe-mirror*, the stripes are created first. Once this is done, the stripes are mirrored. Figure 11.3 illustrates this configuration.

In the RAID 1+0 setup, the disks are first mirrored and then the stripes are created. Figure 11.4 illustrates this configuration.

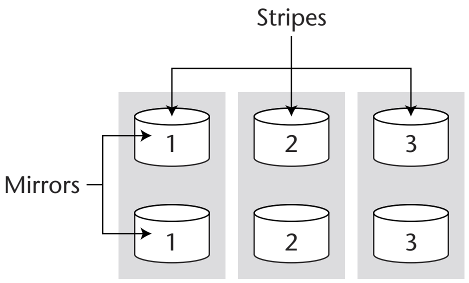


Figure 11.4 RAID 1+0.

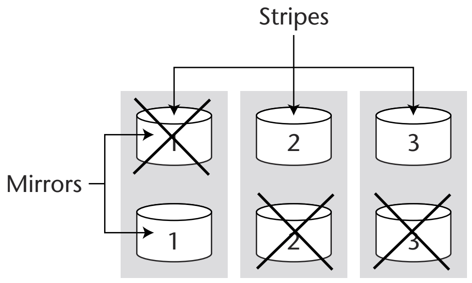


Figure 11.5 RAID 1+0 still working with three failed disks.

Because both of these RAID configurations involve both striping and mirroring, both have the advantages of each of these technologies—namely, improved throughput and redundancy. The primary difference comes into play when a disk fails. If any disk in a RAID 0+1 setup fails, the stripe of which it is a member is taken out of service and we are left with a working stripe and no mirroring. If we lose a disk from each stripe, the data on this volume is inaccessible. In the case of RAID 1+0, the configuration can withstand a greater disk loss. As long as one disk in each set of mirror survives, the device will continue to function. Figure 11.5 illustrates the maximum failure that a RAID 1+0, with the number of disks shown, is able to tolerate and continue working.

Veritas Volume Manager

Veritas Volume Manager has become almost a de facto standard in volume management—especially on the most critical servers. Conceptually, it is much like Solaris Volume Manager and most of the vocabulary will transfer from one product to the other. In other ways, Veritas Volume Manager has its own way of managing disks, its own command set, and its own idiosyncrasies.

Veritas Volume Manager works by loading its own device driver, `vxio`, when the system starts up. This device driver uses information stored in the private regions and uses it to communicate with the underlying hardware. For example, for a mirrored device, it might reissue a write request as multiple writes. By retaining an internal database detailing how devices should handle I/O, the volume manager is able to act as a translator between the physical and virtual devices.

Like Solaris Volume Manager, Veritas Volume Manager supports a number of RAID types, including RAID levels 0, 1, 0+1, 1+0, and 5.

Veritas Volume Manager Terminology

We will start our brief introduction to Veritas Volume Manager (often referred to with the letters VxVM) by comparing its vocabulary to that of Solaris Volume Manager and examining some of its major differences. We will then run through a series of "how to's" to get you started and dissect some Veritas Volume Manager output so that you can analyze the setup of an existing VxVM configuration. In Table 11.6, we compare terminology used by the two volume managers. Figure 11.6 illustrates many of these terms.

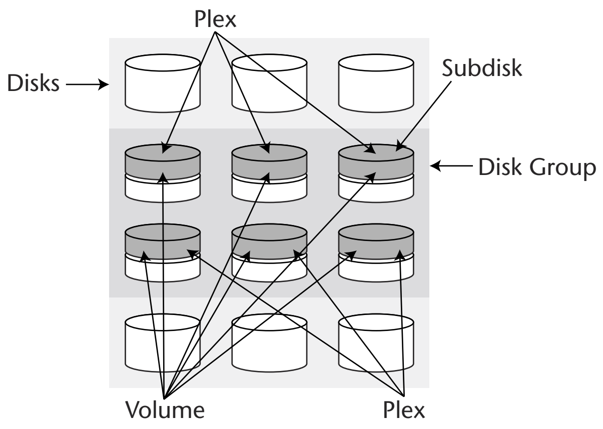


Figure 11.6 Veritas objects.

Table 11.6 Volume Manager Terminology Comparison

SOLARIS VOLUME MANAGER	VERITAS VOLUME MANAGER	DESCRIPTION
Disk set	Disk group	A set of associated disks from which volumes can be created.
Volume	Volume	A virtual disk, created within a disk set or disk group; a SVM volume has a name like d10, while a VxVM volume default volume names are something like vol10, but more meaningful names can be used.
State database replica	Private region	Place where the volume manager stores metadata necessary to keep track of the volumes it controls.
N/A	Public region	That portion of a VM disk that is used for storage.
N/A	VM disk	A disk that has been put under Veritas Volume Manager control (typically named disk##, e.g., disk02).
N/A	Subdisk	A contiguous portion of a disk that does not necessarily correspond to a physical partition.

(continues)

Table 11.6 Volume Manager Terminology Comparison *(Continued)*

SOLARIS VOLUME MANAGER	VERITAS VOLUME MANAGER	DESCRIPTION
N/A	Plex	One or more subdisks located on one or more physical disks; a volume can consist of up to 32 plexes, each of which contains one or more subdisks; a volume with 2 or more data plexes is mirrored; plexes have names like vol10-01.
Physical disk	Physical disk	Any disk attached to your system.
submirror	submirror	One element in a mirrored set.

Installation and Planning

Veritas Volume Manager is installed, as is most commercial software these days, with the Solaris `pkgadd` command. The packages that compose VxVM are:

VRTSvxvm. VERITAS Volume Manager, Binaries.

VRTSvmsa. VERITAS Volume Manager Storage Administrator.

VRTSvmdev. VERITAS Volume Manager, Header and Library Files.

VRTSvmman. VERITAS Volume Manager, Manual Pages.

NOTE Install the packages in the order in which they're listed above to avoid complaints from Solaris about dependencies.

Boot Disk Encapsulation

One notable difference is the way in which Veritas Volume Manager initializes disks when they are put under the Volume Manager's control. Veritas does this in one of two ways—it completely wipes out existing content or it *encapsulates* the disk, preserving the content by making certain changes in the way that the data on the disk is stored.

Encapsulation does the following things:

- Repartitions the disk
- Claims cylinder 0 as `rootdisk-B0` (interferes with VTOC)
- Creates the private region (on top of the public region)
- Creates the public partition
- Preserves all data on the disk

Because many system administrators have had serious problems with their mirrored boot drives (e.g., when they need to recover from a failure or upgrade their OS), Sun has created few blueprints in its Sun Blueprints series to provide an approach that avoids these problems:

"Toward a Reference Configuration for VxVM Managed Boot Disks" (Gene Trantham and John S. Howard). This blueprint helps you avoid the configuration pitfalls which often lead to problems. This document is available from Sun at the following URL: www.sun.com/blueprints/0800/vxvmref.pdf.

"Veritas VxVM Management Software" (Gene Trantham). This blueprint is available at: www.sun.com/solutions/blueprints/0500/vxvmstorge.pdf.

"VxVM Private Regions: Mechanics & Internals of the VxVM Configuration Database" (Gene Trantham). This blueprint provides insights into Veritas' private regions. The approach presented in this paper basically involves mirroring the boot disk and then removing it from rootdg (the root disk group) and reinitializing it. Finally, the disk is added back into rootdg and reattached to the mirror. The overall effect is that the boot disk is no longer encapsulated. Boot disks under the default configuration are overly complex. By making them initialized instead of encapsulated, these complexities are avoided. This also gets around complications resulting from the boot disks not being exact clones of each other. This process is described in the next section, and this blueprint is available at: www.sun.com/solutions/blueprints/0700/vxvmpr.pdf.

Many sites have gotten around difficulties by using Solaris Volume Manager (or its predecessor, Solstice DiskSuite) to mirror the boot drives. The primary problem with this approach is that the system administrators then have to know how to configure and manage two volume managers instead of one.

By spending more time configuring your volume manager in the first place, you will end up with a system that is much easier to manage over the long haul.

A Better Approach to Mirroring Boot Disks

After the Veritas Volume Manager software is installed and the boot disk is encapsulated with `vxinstall`, you can go through a number of steps to reconfigure it so that it is no longer encapsulated. We strongly suggest that you carefully read the Sun blueprint entitled "VxVM Private Regions: Mechanics & Internals of the VxVM Configuration Database" that was mentioned in the previous section before deciding to make this change. However, we believe that you will save yourself a lot of trouble over the long haul. The steps that you must take to make this change are briefly outlined here. Please refer to the blueprint for the full detail.

1. Select the disk to be used as the *mirror* for the root disk and initialize it with the `vxdisksetup` command:

   ```
   # /usr/lib/vxvm/bin/vxdisksetup -i device
   ```

2. Add this disk to the root disk group, rootdg, with the `vxdg` command:

   ```
   # vxdg -g rootdg adddisk rootmirror=device
   ```

3. Attach the mirrors:

   ```
   # /etc/vx/bin/vxrootmir rootmirror
   # vxassist -g rootdg mirror swapvol rootmirror
   ```

4. Disassociate the rootdisk plexes, removing subdisks:

   ```
   # vxplex -g rootdg dis rootvol-01 swapvol-01
   # vxedit -g rootdg -fr rm rootvol-01 swapvol-01
   ```

5. Remove rootdisk from the rootdg:

   ```
   # vxdg -g rootdg rmdisk rootdisk
   ```

6. Initialize the rootdisk (this is what you do with a normal drive), and then add it back to the rootdg:

   ```
   # /etc/vx/bin/vxdisksetup -i device
   # vxdg -g rootdg adddisk rootdisk=device
   ```

7. Attach mirrors in this order:

   ```
   # /etc/vs/bin/vxrootmir rootdisk
   # vxassist -g rootdg mirror swapvol rootdisk
   ```

8. You can now use the `vxmksdpart` command to create partitions on both the rootdisk and the rootmirror.

This method takes some care and planning but, as mentioned earlier, removes the complexities introduced by encapsulated root disks. We encourage you to read the blueprint and follow this well-considered approach to configuring mirrored boot drives.

Booting from a Boot Mirror

The device alias for booting from a Veritas volume will look slightly different from the alias used for a mirror when Solaris Volume Manager is used. The commands for adding the alias to your nvramrc would look roughly like this:

```
nvalias backup /sbus@1,f8000000/esp@1,200000/disk@1,0:a
setenv boot-device disk backup net
nvstore
```

Configuring Your Volumes

The basic steps involved in setting up Veritas volumes include:

- Putting your disks under VxVM control
- Creating disk groups
- Creating volumes on those disk groups

This can be accomplished in any of several ways:

- Using the basic VxVM commands such as `vxinstall`, `vxdg`, and `vxmake`
- Using `vxassist` commands to simplify the configuration
- Using the Veritas Volume Manager Storage Administrator (a GUI tool)

Veritas Volume Manager Commands

Table 11.7 provides a brief description of each of the Veritas Volume Manager commands. To configure Veritas volumes by hand, you will use these commands. Using basic commands to configure your volumes requires the most detail but gives you considerable control over your configuration.

Table 11.7 Veritas Volume Manager Commands

PURPOSE	COMMAND
DISK OPERATIONS	

Initialize Disk	`vxdisksetup -i device`
	`vxdiskadd device`
	`vxdiskadm`

Un-initialize disk	`vxdiskunsetup device`
	`vxdisk rm device`

List disks	`vxdisk list`
List disk header	`vxdisk list diskname`
DISK GROUP OPERATIONS	
Create group disk	`vxdg init diskgroup diskname=device`
Add disk to diskgroup	`vxdg -g diskgroup adddisk diskname=device`
Remove diskgroup	`vxdg destroy diskgroup`
Import disk group	`vxdg import diskgroup`
Deport disk group	`vxdg deport diskgroup`
List disk groups	`vxdg list`
Remove disk from disk group	`vxdg -g diskgroup rmdisk diskname`

(continues)

Table 11.7 Veritas Volume Manager Commands *(Continued)*

PURPOSE	COMMAND
SUBDISK OPERATIONS	
Create a subdisk	`vxmake -g diskgroup sd subdiskname disk,offset,length`
Remove a subdisk	`vxedit -g diskgroup rm subdisk_name`

Display subdiskinfo	`vxprint -st`
	`vxprint -l subdisk_name`

Associate a subdisk to a plex	`vxsd assoc plex_name subdisk_name`
Dissociate a subdisk	`vxsd dis subdisk_name`
PLEX OPERATIONS	
Create a plex	`vxmake -g diskgroup plex plex_name sd=subdisk_name,...`
Associate a plex to a volume	`vxplex -g diskgroup att volume_name plex_name`
Dissociate a plex	`vxplex dis plex_name`
Remove a plex	`vxedit -g diskgroup rm plex_name`
List all plexes	`vxprint -lp`
Detach a plex	`vxplex -g diskgroup det plex_name`
Attach a plex	`vxplex -g diskgroup att volume_name plex_name`
VOLUME OPERATIONS	

Create a volume	`vxassist -g diskgroup -U usetype make volume_name size layout=format disk_name ...`
	`vxmake -g diskgroup -U usetype vol volume_name len=size plex=plex_name`

Remove a volume	`vxedit -g diskgroup rm volume_name`

Display a volume	`vxprint -g diskgroup -vt volume_name`
	`vxprint -g diskgroup -l volume_name`

Change volume attributes	`vxedit -g diskgroup set field=value volume_name`
	`vxvol field=value... volume_name`

Table 11.7 Veritas Volume Manager Commands *(Continued)*

PURPOSE	COMMAND
VOLUME OPERATIONS	

Resize a volume	`vxassist -g diskgroup growto volume_name new_length`
	`vxassist -g diskgroup growby volume_name length_change`
	`vxassist -g diskgroup shrinkto volume_name new_length`
	`vxassist -g diskgroup shrinkby volume_name length_change`
	`vxresize -F ufs -g diskgroup volume_name new_length`
	`vxvol -g diskgroup set len=value volume_name`

Change volume read policy	`vxvol -g diskgroup rdpol round volume_name`
	`vxvol -g diskgroup prefer volume_name preferred_plex_name`
	`vxvol -g diskgroup rdpol select volume_name`

Start/stop volumes	`vxvol start volume_name`
	`vxvol stop volume_name`
	`vxvol stopall`
	`vxrecover -sn volume_name`

List unstartable volumes	`vxinfo [volume_name]`

Mirror an existing plex	`vxassist -g diskgroup mirror volume_name`
	or
	`vxmake -g diskgroup plex plex_name sd=subdisk_name`
	`vxplex -g diskgroup att volume_name plex_name`

Snapshot a volume	`vxassist -g diskgroup snapstart volume_name`
	`vxassist -g diskgroup snapshot volume_name new_volume`

Add a log to a volume	`vxassist -g diskgroup addlog volume_name`

Using vxassist

The vxassist command provides for simplified configuration of Veritas volumes by requiring a minimal interaction with the user. The vxassist commands, included in Table 11.8, make use of a file containing defaults and therefore require almost no information from the user. The file containing the defaults is /etc/defaultvxassist.

With vxassist commands, you can create and modify simple volumes that are, of course, completely compatible with volumes set up manually. vxassist finds space for and creates volumes and mirrors, extends or shrinks volumes, provides for online backups, and estimates the maximum size for a new volume.

Table 11.8 The vxassist Commands

OPERATION	COMMAND
Create a volume with the specified name and length	`vxassist [options] [-b] make volume_length [attribute...]`
Create a new mirror (or plex) and attach it to the volume	`vxassist [-b] [options] mirror volume [attribute...]`
Move subdisks within the named volume off the excluded storage specified on the command line	`vxassist [options] [-b] move volume !storage-spec... [attribute...]`
Increase the length of the named volume to the length specified by *newlength*	`vxassist [options] [-b] growto volume newlength [attribute...]`
Increase the length of the named volume to the length specified by *lengthchange*	`vxassist [options] [-b] growby volume lengthchange [attribute...]`
Decrease the length of the named volume to the length specified by *newlength*	`vxassist [options] shrinkto volume newlength`
Decrease the length of the named volume by the length specified by *lengthchange*	`vxassist [options] shrinkby volume lengthchange`
Create a temporary mirror and attach it to the named volume	`vxassist [options] [-b] snapstart volume [attribute...]`
Create a new volume with the temporary mirror as its one plex	`vxassist [options] snapshot volume newvolume`
Wait for an attach to complete on the named volume	`vxassist [options] snapwait volume`

Using the Storage Administrator

The Veritas Volume Manager Storage Administrator (VMSA) provides a GUI interface for configuring and managing your volumes. The obvious benefit is that this makes it easier for the user to picture how the volumes are being configured. VMSA is pictured in Figure 11.7.

System Changes

Once Veritas Volume Manager is running on your system and configured, you will notice numerous changes in important system files. In your `/etc/system` file, you will notice that lines like these have been added. These are line load drivers needed by Veritas Volume Manager and should not be removed from this file.

```
forceload: drv/vxio
forceload: drv/vxspec
```

You will notice that the lines in your `/etc/vfstab` file no longer look the way they did before. Lines of this type:

```
/dev/dsk/c0t2d0s1   -        -        swap    -     no      -
/dev/dsk/c0t2d0s0   /dev/rdsk/c0t2d0s0      /     ufs     1   no     -
```

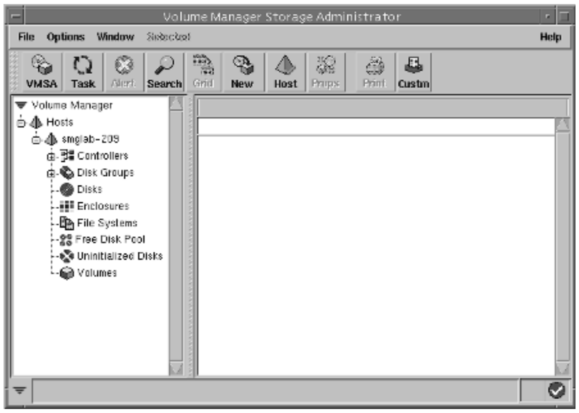

Volume Manager Storage Administrator
File Options Window Selected Help
VMSA Task Alert Search Grid New Host Props Print Custm
Volume Manager
Hosts
smglab-209
Controllers
Disk Groups
Disks
Enclosures
File Systems
Free Disk Pool
Uninitialized Disks
Volumes

Figure 11.7 Volume Manager Storage Administrator.

will have been replaced by lines such as these:

```
/dev/vx/dsk/swapvol  -       -       swap    -      no      -
/dev/vx/dsk/rootvol  /dev/vx/rdsk/rootvol    /     ufs     1   no    -
```

The changes will also be reflected in your `df` command, of course, which will look something like this:

```
Filesystem          kbytes    used   avail capacity Mounted on
/dev/vx/dsk/rootvol 12249580 6326058 5801027 53%     /
```

Veritas Volume Manager commands can be found in the `/etc/vx/bin` directory.

Understanding vxprint Output

If you want to see details on how your volumes are configured, Veritas Volume Manager will provide all the output you could want and then some. Various forms of the `vxprint` command can be used to display different aspects of your volume setup. We'll look at two of them in the following sections.

vxdisk list

The `vxdisk list` command will provide a list of the disks that have been put under Veritas Volume Manager's control. The output from this command will look something like this:

```
# vxdisk list
DEVICE    TYPE   DISK    GROUP     STATUS
c0t2d0s2  sliced c2t12d0 rootdg    online
c0t3d0s2  sliced c2t13d0 rootdg    online
c1t0d0s2  sliced c1t0d0  finapps00 online
c1t1d0s2  sliced c1t1d0  finapps00 online
c1t2d0s2  sliced c1t2d0  finapps00 online
c1t3d0s2  sliced c1t3d0  finapps00 online
c1t4d0s2  sliced c1t4d0  finapps00 spare
```

The output above shows two disk groups and seven disks (one of which is designated as a spare). Six of the disks are online.

vxprint -hrt

In the `vxprint` command shown below, you see a nice summary of Veritas objects. Notice that the eight lines following the name of the disk group contain the headers for the data that follows. For example, the `DM` line tells you that the fields in each subsequent `dm` line contain the name of the disk, the actual device, the type of device, the lengths of the private and public regions (in sectors), and the state of the device.

```
# vxprint -hrt
Disk group: rootdg

DG NAME         NCONFIG      NLOG     MINORS   GROUP-ID
DM NAME         DEVICE       TYPE     PRIVLEN  PUBLEN   STATE
RV NAME         RLINK_CNT    KSTATE   STATE    PRIMARY  DATAVOLS  SRL
RL NAME         RVG          KSTATE   STATE    REM_HOST REM_DG
REM_RLNK
V  NAME         RVG          KSTATE   STATE    LENGTH   USETYPE
PREFPLEX RDPOL
PL NAME         VOLUME       KSTATE   STATE    LENGTH   LAYOUT
NCOL/WID MODE
SD NAME         PLEX         DISK     DISKOFFS LENGTH   [COL/]OFF DEVICE
MODE
SV NAME         PLEX         VOLNAME  NVOLLAYR LENGTH   [COL/]OFF AM/NM
MODE

dg rootdg       default      default  0        766546412.1025.www-s1

dm rootdisk     c0t0d0s2     sliced   3023     4122719  -

sd rootdiskPriv -            rootdisk 3334464  3023     PRIVATE   c0t0d0
ENA

v  cache        -            ENABLED  ACTIVE   263088   fsgen     -
ROUND
pl cache-01     cache        ENABLED  ACTIVE   263088   CONCAT    -
RW
sd rootdisk-03  cache-01     rootdisk 3859631  263088   0         c0t0d0
ENA

v  rootvol      -            ENABLED  ACTIVE   3334464  root      -
ROUND
pl rootvol-01   rootvol      ENABLED  ACTIVE   3334464  CONCAT    -
RW
sd rootdisk-B0  rootvol-01   rootdisk 3334463  1        0         c0t0d0
ENA
sd rootdisk-02  rootvol-01   rootdisk 0        3334463  1         c0t0d0
ENA

v  swapvol      -            ENABLED  ACTIVE   522144   swap      -
ROUND
pl swapvol-01   swapvol      ENABLED  ACTIVE   522144   CONCAT    -
RW
sd rootdisk-01  swapvol-01   rootdisk 3337487  522144   0         c0t0d0
ENA
```

This output shows a system with a single disk group (`rootdg`), a single disk (`root-disk`), three volumes (`cache`, `rootvol`, and `swapvol`), three plexes (one on each volume), and five subdisks. Table 11.9 provides a definition for each of the object descriptors.

Table 11.9 Object Descriptors

DESCRIPTOR	MEANING
dg	Disk group
dm	Disk
rv	Replicated volume group
rl	Rlink (used in Veritas Volume Replicator)
v	Volume
pl	Plex
sd	Subdisk
sv	Subvolume
dc	Data change object
sp	Snap object

Summary

How many times have you been asked whether you can enlarge a partition? Without volume management, the answer is a long-winded explanation of how file systems can be backed up, disks repartitioned and file systems reloaded—at a significant cost in terms of time and trouble. With volume management, the answer is much more likely to be “Sure!” followed by a quick check to see how many other competing tasks are in your queue for the afternoon and a check of how much unused disk space is available.

Without volume management, you are likely to find that you have free disk space where you don’t need it and inadequate space where it’s most needed. With volume management, you can maximize data availability as well as adjust disk space to be flexible to changing demands—and without the tedious work of moving file systems around. Your users *still* won’t throw anything away, but managing free disk space will slide from the top of your task list.

In this chapter, we explored the vocabulary of volume management. We provided some basic information about the capabilities and the command set for Solaris Volume Manager and Veritas Volume Manager. Because the administration guide for each of these tools is several hundred pages long and this chapter is nowhere near that long, we urge you to read additional material on these tools (to this end we provide you with some references in the back of the book).

Volume managers give you a considerable degree of flexibility over the control of your disk space and high availability—but at a cost in terms of complexity (just consider how much detail was presented in this short introduction). In essence, you can use a large number of disks as if they were one big chunk of disk space and then break the space up to match your needs. Options for redundancy allow you to build file systems that can literally be available 24 × 7. At the same time, configuring and managing your disk space requires considerable familiarity with the volume manager of your choice.

CHAPTER

12

Automating Everything . . . Well, Almost!

The essence of Unix in general, and Solaris in particular, is how flexible and extensible it is. Extending the OS with software that you create from scratch or download from archives on the Internet is the best way to tame your systems and your job. Scripts you build or borrow and tools you acquire will help you manage user accounts, control file systems, and monitor applications and system performance. They will also help you configure services and extract, from the extensive amount of log data, the vital statistics that will tell you how your network is doing and what aspects of it require your immediate attention.

From our perspective as long-time system administrators, the answer to the question “How much should we automate?” is a resounding “Almost everything!” Any task involving more than a few steps that you’re going to perform more than half a dozen times is a candidate for automation. The second question to ask in regard to automation is “How?” Quick and dirty scripts have their place; they can dramatically reduce the drudgery associated with something you have to do a number of times in a row. For the tasks you perform only now and then or automatically under the control of `cron`, however, anything less than carefully written and tested scripts will, in the long run, prove inefficient and difficult to maintain. We give you pointers in this chapter not only on what to automate, but how to ensure that your scripting efforts will pay off.

Your Fill of Scripting and Then Some

In our experience, the predominant activity of system administration is not installing systems, backing them up, or creating and deleting user accounts. It's not tracking hackers, formatting disks, compiling software, or even answering questions. It's *writing scripts*—scripts that manage tasks like these, along with a myriad of others.

Anything you can do interactively, with care, you can script.

TIP Our advice is to script anything you expect you'll have to do more than half a dozen times—or that you do so rarely that you're not likely to remember the commands if you don't commit them to disk.

If you prepare scripts for your users, they may be able to do a lot for themselves that you might otherwise have to do for them. Not only will this make your jobs easier; it will make their jobs easier, as well. Even the most demanding users are likely to prefer taking care of themselves to waiting for you, with your long list of tasks, to get around to taking care of their latest requests.

Benefits of Scripting

Another benefit of scripting is that, with a good collection of scripts, you can more easily turn your operation over to someone else. Each of the authors has, at times in their careers, inherited a complex network and application setup to administer. Were it not for the body of knowledge contained in our predecessors' scripts, the first few months on those jobs would have been miserable. With them, it was still stressful—but manageable.

If your operation is easy to turn over to someone else, you'll have an easier time accepting a promotion when an irresistible opportunity comes along. If you're in the perfect job already, you'll find that you'll benefit from your efforts as much as your replacement would have. Think of your scripts as investments that pay off every time you do a job in a tenth of the time it would otherwise take. Plan for many of your scripts to serve as modules that you can use and reuse as new challenges arise and new solutions are needed.

We cannot overstress this issue. Writing scripts *will* save you a lot of time and trouble. Any scripts you write will almost always be:

- Faster than running the same commands by hand
- Far less tedious than running the commands by hand
- Reliable and consistent (you might forget a critical step in a complex set of commands)
- Easily scheduled to run at routine, less busy times (say when you're asleep)

Scripting Discipline

Scripting does involve a certain discipline. Like any other software, the scripts you write should be free of bugs and should be written in a style that makes them easy to maintain (i.e., to modify to adjust to changing circumstances).

It almost doesn't matter what scripting language you use, though we recommend that you become adept at programming in several languages. Among these are the Bourne shell and, optionally, its derivatives (Korn and Bash shells), the C shell, and Perl.

TIP Consider writing time-intensive routines in C; compiled programs almost always run considerably faster than interpreted shell scripts.

To make the most of your scripting efforts, you should immediately—if you have not already done so—adopt certain conventions in your scripting style. These include naming and storage conventions, as well as coding and formatting style. Over time, these conventions will save you a lot of time. Following are some suggestions that have served us well:

- Create a `howto` directory in which you will store examples of commands that you rarely use—especially those that took you a while to get right—and for which there are no examples in the man pages.
- Develop a migration scheme in which your in-progress scripts are clearly distinguished from those that you've thoroughly tested.
- Put your trusted scripts in a `bin` directory—in your personal account folder if only you are to use them, or in the `/usr/local/bin` if they are to be made available to other members of your staff or to users.
- Use a script-naming scheme that distinguishes between scripts that are to be executed directly and those that are used only by other scripts. This can be as simple as using a `thisName` convention for scripts that are meant to be executed directly and `this_name` for subsidiary scripts.
- Use a naming scheme that makes it easy to remember what each script does. Take advantage of abbreviations such as *mk* (make) and *ls* (list); *rm* and *mv* will have obvious meanings because of the corresponding Unix commands. You probably won't enjoy recreating a script you well remember writing and testing simply because you can't remember where you put it or what you named it.
- Initialize important parameters at the top of your scripts so that you'll never have to go hunting through the lines of code to change them.
- Include parameter checks and usage statements in each script to help ensure that scripts are not used incorrectly (and that *you* don't have to analyze the code to remember how to use them!).
- Use a consistent programming style.

- Use cut-and-paste or script templates to save yourself time and effort and avoid annoying syntactical glitches.
- Create simple script utilities for repetitious code segments.
- Decide on a useful organizational style. Will you store scripts by project or by type? Any organizational tricks that you can develop will probably end up saving you lots of time.
- Clean up your scripting directories on a routine basis to get rid of abandoned efforts and intermediate copies. This increases the prominence and usability of what is left.
- Back up your scripts from time to time or maintain at least retain printouts of your scripts in a binder.

The Finer Points of Scripting

System administrators are presented with an almost endless stream of opportunities to become experts. The tools and languages available to help with our jobs and never-before-seen problems that crop up from time to time keep us from ever settling into a routine. Even so, there is an essential core of scripting skill and technique that, when mastered, will allow us to stay ahead of utter chaos. Our take on these skills and techniques is briefly covered in the following sections. Mastering any of the topics presented here will take a lot more information and practice than we have room for in this chapter. We suggest that you take advantage of the many fine texts available on scripting and that you master several languages—including Perl and at least one shell.

Dealing with Limits

Sooner or later, if you haven't already done so, you're going to run into limits. Your standard method of solving a problem is suddenly not going to work. For example, perhaps a tool that modifies links in HTML files—for example, to change them from fixed to relative—will be asked to use one more pattern than it can handle and will issue an error message and stop. In our testing, the `foreach` command in the C shell balked and issued the error "Too many words" at just over 1,700 items in its expanded list. This is because the `sed` utility can only handle so many patterns before it balks. Our `sed` command gave up the ghost and told us `Too many commands` with just under 200 patterns in our file.

As another example, our `awk` script refused to budge when we passed it data with 100 fields. The `awk` tool can only handle so many fields before it just says `Too many fields` and quits. Elegant workarounds to problems like these are few and far between.

You can use regular expressions as one possible workaround for these kind of limits. If you can reduce the number of patterns that you are trying to match by using regular expressions instead of literals, you may stay below the tool's threshold. At other times, you can break the problem into pieces and build your output file gradually rather than in a single step.

Often the best workaround is to use another tool. A newer member of the awk family, such as gawk or nawk, might be able to work with a larger file. A compiled language, such as C or C++, will offer little resistance when called into battle against large problems, but will probably take considerably more effort in preparation. Finding ways to work against limits is one of the signs of a seasoned sysadmin.

Using the Right Tool for the Job

One thing that can and has, many times, been said about Unix systems is that there is always more than one way to solve a problem. Most of the time, there's even more than one *good* way to solve a problem. However, each of the tools and languages that compose our systems has its particular strengths and weaknesses. Though you might be able to use either sed or tr to solve a problem, one might have decided advantages over the other. The following list describes each of the major tools as well as when each is useful.

- The *Bourne shell* is one of the most popular shells. One reason is that it is ubiquitous; you will not find a Unix system without it. Another is that Bourne shell scripts will run with newer shells in the same shell family—such as the Korn shell and bash (the Bourne Again shell).
- The *C shell* is a nice shell with easy-to-use features such as the foreach command and interactive file completion. However, the C shell has fallen out of favor with many sysadmins because of an earlier bout of security problems and because it lacks many of the features found in more modern languages such as Perl (e.g., reading multiple files). However, it has many nice features and is extremely useful for quick and dirty scripts.
- An easy favorite, the *Perl* language is capable of being extremely terse. It makes extensive and fluid use of regular expressions. Once familiar with Perl, you're likely to use it almost exclusively. It provides numerous features of other Unix tools—for example, sed and awk. Freely available, and popular partly due to its use in cgi scripts for Web sites, Perl promises short development time for those who know it. It also includes a number of features missing from older languages—good array handling, string functions, and, of course, *impressive* regular expressions. It is very straightforward in its file handling.

  Perl is used extensively both for system administration and Web site management because of its efficient syntax and the ease with which it handles any data that can be described.
- *Expect* is the tool to use when you can't be there. Expect will interact with some utility that expects real-time answers and provide these answers on your behalf. You basically prepare a script of what to expect and how to respond to prompts.
- *C* and *C++* are two of the more popular languages in use today and are usually available on Solaris systems, whether as commercial software or the GNU versions. When you need speed, it's hard to beat the running time of a compiled program.

- *Java* is used considerably in Web development these days, using servlets with JavaServer Pages (JSP) and using tools such as Tomcat (the Java Servlet and JSP container developed by the Apache Foundation). *JavaScript*, a scripting language developed by Netscape, is also used quite heavily.
- `awk` is a pattern-matching language, great for grabbing the *n*th column in a delimited file, and good for writing simple filters. To grab the third field in the `passwd` map using awk and NIS, you would use `enter: ypcat passwd | awk -F: '{print $3}'`.
- `sed`, the Unix stream editor, is good for processing changes in a file. It is generally used inline (e.g., `cat file | sed "s/this/that/"`), but can also be used with a prepared file of expressions.
- `grep` is another tool for use with regular expressions. It provides one of the easiest ways to select and deselect patterns. Contrast `grep` with `egrep` and `fgrep`, which allow you to search for words or word patterns with a file of target patterns.
- `tr` is a tool for translating characters or strings to other characters or strings. A common example of this command is `cat file | tr "A-Z" "a-z"`. It changes all uppercase letters to lower case. Similarly, `cat file | tr -d "\015"` gets rid of carriage returns in a DOS text file (turning it into a Unix text file), while `cat file | tr "\015" "\012"` turns Macintosh text file endings into Unix text file endings by replacing carriage returns with new lines.
- `wc` is a command that counts lines, words, or characters. Combined with grep, for example, `wc` can tell you how many times a particular pattern appears in a file. However, other commands also provide counts as well. The `uniq -c` command has proven to be just the right tool for many applications because it provides counts on how often each unique string appears.

Good Scripting Practices

There are many simple techniques that can make script writing more reliable and less frustrating. In the following sections, we provide some guidelines that have worked for us over the years.

Build up to Complexity

One recommendation we make, especially to those just getting started in the business of automating their work, is to start simply and build up to complexity. When starting a script that is destined to be complicated it is good practice to first build the control structure. Later, you can fill in the commands. Inserting simple comments at each level in the looping or case structure will help you verify that it's working properly before additional details weigh down the process. You could start with something as simple as this:

```
foreach FILE (`find ./* -name *.dat -print`)
  echo $FILE
end
```

Once you have verified that you're picking up the correct files, you can start adding the commands to process these files. But wait! Another idea that we've used on occasion is to next `echo` the commands that you will be using. You then verify that the generated commands appear to be proper before you remove the `echo` command and allow the commands to be executed in your first trial. For example, you might change the preceding code to this:

```
foreach FILE (`find ./* -name *.dat -print`)
  echo bldTable $FILE `date +%y%m%d`
end
```

This loop would display the command that will be used to process each file when the script is made operational.

Keep Skeletons in Your E-Closet

When you've put the time into building up a control structure that lends itself to reuse, store the code in this basic form for reuse. We have found that many of the scripts that we have developed over the years have had the same basic form. Starting from a file that contains the "shebang" line and the commands for checking arguments and opening a data file might save you from writing these lines over and over again.

If, for example, you're going to be building Web pages with Perl, you might start with a code skeleton that looks like this and add your processing a little at a time, while checking the end result in a browser:

```
#!/usr/bin/perl

$datafile="datafile";

open(DATA, $datafile) || die("Cannot open $datafile!");
@indata=<DATA>;
close(DATA);

print "Content-type: text/html\n\n";
print "<HTML><BODY>";

foreach $record (@indata)
{
 chop($record);

[add processing here]

 print "<BR>\n";
}

print "</BODY></HTML>";
```

This skeletal code reads a data file, storing the contents in an array. It then processes each record in the array in some way. Before and after this processing, however, it adds the commands required to identify the end result as an HTML file. In between each processed record, it inserts a <BR> to cause a break at the end of the line.

By following techniques such as these, you're far less likely to run into problems with mismatched begins and ends or similar problems that otherwise consume quite a bit of time. In the following Bourne shell code, you can see how we've used this technique to display the `mv` and `ln` commands before allowing this script to affect the actual files. After you're confident that you've constructed your file names correctly, you can remove the `echo` commands.

```
#!/bin/sh
PATH=/usr/ucb:/usr/local/bin:$PATH ; export PATH
me=`basename $0 2>/dev/null`
year=`date +%Y`
mo=`date +%m`
umask 022
echo mv /www/index.html /www/index.html$$
echo mv /usr/tmp/$me/new.html /www/$me/$year-$mo.html
echo ln -s /www/$me/$year-$mo.html /www/index.html
```

Test Your Scripts with Increasing Amounts of Data

Another important rule of thumb for developing scripts is to test them with increasingly realistic amounts of data. Early tests should be run with a *very* small sample, both because this allows you to closely examine the results before proceeding and because you'll get these results quickly. A process that runs an hour or more and then proves to have generated bogus results can waste an hour or more of your time.

Check Files before Trying to Use Them

Scripts that you create should also be well behaved. There are certain characteristics of a well-behaved script that we would like to share. For one, well-behaved scripts ensure that a file exists before they try to use it. If the file doesn't exist, a well-behaved script will generate an error and return (exit). If the file should have been included as an argument, a well-behaved script will issue a usage statement, trying to assist the user, and exit. Table 12.1 shows examples for three scripting environments.

While we're opening files, we should not fail to mention that Perl has specific syntax for opening files for reading, writing, and appending. While the command in Table 12.1 opens the data file and exits if it cannot, the command below opens the data file for a particular operation:

```
open(DATA,"<$file") or die "cannot open $file for read";
open(DATA,">$file") or die "cannot open $file for write (overwrite)";
open(DATA,">>$file") or die "cannot open $file for append";
open(DATA,"+>>$file") or die "cannot open $file for read/append";
```

Table 12.1 Checking for the Existence of a File

#!/bin/csh	#!/bin/sh	#!/usr/bin/perl
if (! -f $1) then	if [! -f $1]; then	open(DATA, $datafile) ||
echo "cannot find	echo "cannot find	die("Cannot open
$1"	$1"	$datafile");
exit	exit	
endif	fi	

Check for the Correct Number and Type of Arguments

Well-behaved scripts should also check the number of arguments provided at run time if the arguments aren't optional and should issue an understandable usage statement if there is a problem. Table 12.2 illustrates how to do this in each of the three scripting environments.

There are other ways to do this checking as well. For example, in the Bourne shell, you might simply check whether the first argument has any value. If it does not, the syntax below will compare one empty string with another.

```
#!/bin/sh

if [ "$1" = "" ]; then
    echo "Usage: $0 <site#>"
    exit
fi
```

Table 12.2 Checking the Number of Arguments

#!/bin/csh	#!/bin/sh	#!/usr/bin/perl
if ($#argv != 2) then	if [$# != 2]; then	$numArgs = $#ARGV + 1;
echo "usage:	echo "usage: $0	if ($numArgs lt 2) {
$0 <yr> <mo>"	<yr> <mo>"	print "usage: $0
exit	exit	<yr> <mo>\n";
endif	fi	}

An alternate way to check for a single argument in Perl uses the `shift` command, as shown here:

```
my $siteNo = shift
    or die "Usage: $0 <site#>\n";
```

This code attempts to assign the value of the next argument in the argument list to `$siteNo`. If there is no such argument, the script exits with a usage statement. The following Perl code reports how many arguments were provided on the command line and lists them one at a time. This script also contains its name in the comments at the time.

NOTE Many Perl programmers use "pl" as the file extension of their Perl scripts to differentiate them from other file types.

```
#!/usr/bin/perl
#-----------------------------------------------
#  PROGRAM:  argv.pl
#-----------------------------------------------

$numArgs = $#ARGV + 1;
print "$numArgs arguments provided on the command line\n";

foreach $argnum (0 .. $#ARGV) {
   print "$ARGV[$argnum]\n";
}
```

It is also good practice for any script that is to be called by another script to exit with an error code (i.e., `exit 1`) if it exits prematurely so that the calling script can determine that a failure occurred. Although you may never use these return codes, you might use them if you ever call one of these scripts from another script. In this case, knowing whether the called script was completed properly or ended abnormally might be important in determining how the calling script should react. The following code snippet shows a simple example of a script that returns a 1 if it was not run from the proper directory. Notice that the `$?` (`/bin/sh`) variable contains the exit code.

```
#!/bin/sh
if [ `pwd` != "/usr/local/bin" ]; then
    echo "This script can only be run from /usr/local/bin"
    exit 1
fi
$ ./sh8
This script can only be run from /usr/local/bin
$ echo $?
1
```

TIP **Include error messages and usage statements in your scripts—especially if they will be used infrequently. You'll save yourself and anyone else who uses the scripts a lot of time. They won't have to read your code to figure out how to use it.**

Manage Your I/O

Well-behaved scripts take responsibility for their I/O. Before crafting a script, you should decide what output you want users to see and what you prefer to hide from them. Many `cron` scripts, for example, send their output to `/dev/null`, not because there's anything wrong with the output that is produced, but because being reminded once a day that a script that runs once a day has been completed successfully is of little value—especially if it lulls users into the bad habit of ignoring email from the system.

Shell redirection allows you to be creative with output. You can store it in a file or make it disappear altogether. You can append it to an existing log file or overwrite it each time a script is run.

To fully understand the syntax used by redirection, you should understand how file descriptors work. Every file in use by a program has a file descriptor (e.g., 5) that the program uses when accessing that file. A Unix shell uses file descriptors to manage I/O with the user as well. These include standard in, standard out, and standard error and are associated with file descriptors 0, 1, and 2, as shown in Table 12.3. Table 12.4 provides the commands for redirection in the various shells.

Consider sending output to `/dev/null` or to temporary files when it might distract users from what they should be paying attention to. If you redirect output to a temporary file and issue a message pointing to the file's location, the user can read the file if and when he or she wants to.

You can use redirection to simplify many tasks. Here's a nice way to add text to the bottom of a file without opening it in an editor.

```
cat >> myfile
This is text that I want to add to
the bottom of myfile just like this.
CTRL-D
```

Table 12.3 File Descriptors

FILE DESCRIPTOR	REFERS TO . . .
0	Standard input (generally the keyboard)
1	Standard output (generally your display)
2	Standard error (generally your display)

Table 12.4 Redirection

	#!/bin/csh	#!/bin/sh
Write to a file	`>`	`>`
Append to a file	`>>`	`>>`
Read from a file	`<`	`<`
Read from standard in until specified word is encountered	`<< word`	`<< word`
Append standard error		`2>>`
Discard standard error		`2>/dev/null`
Combine standard out and standard error	`>&`	`>&2`
Append standard out and error	`>>&`	`cmd >> filename 2>&1`

Don't Overuse Pipes

As much as the use of pipes is one of the things that has made Unix what it is, overuse of pipes comes at a cost in terms of performance. If you're running a command interactively and it will only take a matter of seconds to complete, optimization is a waste of time. If you're preparing a script and dealing with large amounts of data, on the other hand, every process that you avoid creating is likely to shorten the run time of your script.

For example, why is this snippet of code:

```
grep "^Subject: " /var/mail/ram | sort | uniq -c
```

better than this one?:

```
grep "^Subject: " /var/mail/ram" | sort | uniq | wc -l
```

The answer is simple—we've created one less process. For some tasks, the difference can be an order of magnitude in how long the overall command takes to process. Similarly, this command:

```
if [ `pgrep $PROC` ]; then
```

is more efficient than this:

```
if [ `ps -ef | grep $PROC | grep -v grep | wc -l` ]; then
```

and for the same reason. The `pgrep` version of the command is also more readable.

Create Scripts Interactively

One way to combine testing and script buildup is to run your commands interactively, as shown in the following example, and then, when you're satisfied with the results, cut and paste the commands into a file. This technique generally works better with simple scripts. Once you've put the effort into typing the commands for a useful function, save them in the form of a script, give them a name you can remember, and store them in your /bin or /howto directory. The small effort involved might save you time someday when you need as much of it as you can find. For example, if you type the following commands interactively:

```
spoon% foreach file (`ls /logs/*`)
? gzip $file
? echo $file compressed
? end
```

Your cut-and-pasted script will have to be modified slightly to look like this:

```
#!/bin/csh
foreach file (`ls /logs/*`)
  gzip $file
  echo $file compressed
end
```

Don't overlook the utility provided by Sun's cmdtool windows for saving the results of your processing. A mouse right-click brings up, among other things, the option to store or clear the contents of the log file. This can provide a useful way to recover commands that worked especially well—a little while ago. Because many of your efforts might initially be hit or miss, you might not always recognize a good approach to a problem until after you've tried it again. It's nice to know that these gems of scripting wisdom can still be retrieved—either from your command history or from a cmdtool log.

Once you've captured your gem, another option (besides depositing the commands into a script) is to create an alias—at least for the single liners—and store it in your shell dot file (e.g., .cshrc), as follows:

```
alias usage "du -k | sort -rn | more" <-- csh
alias usage="du -k | sort -rn | more"; export usage <-- sh family but
not /bin/sh
```

Manage File Permissions

Whenever you're scripting for general use, be careful with permissions. An output file owned by root might not allow appending to a file if the next person to run it is a normal user. If you do your testing as root, be sure to watch for potential side effects. Running your tests as a normal user will help ensure that they'll work for other normal users.

Make Scripts General Purpose Whenever Possible

Another good practice is to write your scripts to be as general as possible. You might be able to use them in a variety of situations. An extremely simple script can be thrown into many situations to save time when a common task, such as counting the repetitions of the same string or pattern is important.

Provide for Easy Debugging

Another good idea is to leave your debugging lines in your code, but to comment them out; or, even better, provide a verbose or debug mode with lines like the following chunk of Perl, so that you can run the script with helpful comments if you need to:

```
if ($verbose) {
  print $_;
}
```

The syntax shown below will turn debugging mode on for a Bourne shell script, and display each line in the script as it is being run, if the user adds the word `-debug` as an argument when he or she invokes the script:

```
if [ "$1" = "-debug" ]; then
    set -x
fi
```

Randomize Temporary Files

Temporary files can be made to be almost certainly unique by adding `$$` (the process ID) to the end of them (`procid` of the current shell) in the Bourne and C shells. The likelihood of a conflict is exceedingly small, and this practice will save you from being concerned about whether previous output may interfere with your results.

Make a practice of storing results from potentially time-consuming operations rather than running them again. For example, note how the `set` command in the example shown here is both easier to understand and more efficient than putting the two `grep...wc` commands and the `expr` command inside of the two `echo` commands:

```
set SUBS = `grep "Subject: subscription" subfile | wc -1
set DROPS = `grep "Subject: remove" subfile| wc -1
echo "$SUBS newsletter subscriptions"
set PCT = `expr $DROPS \/ $SUBS \* 100`
echo "$PCT % drops"
```

Beware of Exceptions

As is the rule in any programming situation, carefully watch those *endpoints*—one of the most troublesome parts of scripting is dealing with the first and last instances of anything. Be careful of these as you write and test your scripts. For example, in counting lines that are the same, the first line is an exception, because there *are* no preceding lines, and the last line is an exception, because you have to remember to print the count accumulated thus far (usually done when you encounter a different line).

Watch out for Script "Droppings"!

Take care to ensure that your scripts will not run into problems if they are executed more than once. Getting rid of temporary files at the ends of scripts is generally a good idea, especially if the script appends data to these files. If it is important that a script not be run more than once (e.g., not more than once a month), be careful to implement that restriction in your logic. Make sure that you can easily and reliably identify when a script has been run.

Add Comments, but Don't Overdo It

Add comments to your scripts, but not excessively. A `foreach DAY` (Monday Tuesday Wednesday . . .) command does *not* need a corresponding comment that says `# for each day of the week`—it's obvious. On the other hand, a complex regular expression such as `(m/^From\s(\S@\S)\s(.*)/)` deserves a comment `# At beginning of email message...`.

Test Your Scripts

Don't fail to test your scripts. If necessary, create clones of data files and make sure you get the results you expect before you turn them loose on important files. There is no worse feeling than realizing that your brand-new clever script has just demolished a critical data file because a command was just a little wrong. At the same time, be careful, *and* forgive yourself for these mistakes. We all make them.

Running Scripts at Regular Intervals

There is nothing like `cron` when it comes to running your scripts at regular intervals. Using `cron`, you can schedule jobs to run at any minute of any day. You can run a script as infrequently as every time January 8 falls on a Tuesday (approximately once every 7 years). Most `cron` scripts are run once a day, once a week, or once a month.

Running your scripts through `cron` is easy. The `cron` fields are listed in Table 12.5.

Table 12.5 The Fields on a crontab File

FIELD	RANGE	EXAMPLE
Minutes	0-59	10
Hours	0-23	9
Day of month	1-31 (but watch out!)	8
Month	1-12	7
Day of week	0-6 (Sunday to Saturday)	6

The `cron` entry corresponding to the examples in Table 12.5 would be listed as follows:

```
10 9 8 7 6 /usr/local/bin/runme 2>&1 | mail sysadmin
```

This would cause the script `runme` to run at 9:10 on July 8 if it's a Saturday. Any field can include more than one value (e.g., 9, 21) or an * representing any possible value. Unfortunately, there isn't any easy way to specify that you want a script to run on the last day of the month, so monthly scripts are usually run on the first, shortly after midnight. If you needed to run a script on the last day of the month, you could schedule it to run every night and quickly exit if it isn't the last day of the month.

If you want a script to run shortly after midnight, but to wrap up log files and such from the day before, start the job shortly before midnight and insert a `sleep` command, as shown in the following example. This will cause it to wait the appropriate amount of time before it starts to process the previous day's files.

```
#!/bin/sh
YR=`date +%Y`
MO=`date +%m`

sleep 120
```

If this script is run at 23:59, it will pick up the date of the day that is ending and hold it for use after the brief period of sleep. This is a good technique if you want to be sure to include a complete day's data.

Updates to the `cron` files should always be made using the `crontab -e` command. This gives `cron` a chance to balk on your syntax if you make a mistake (e.g., omitting one of the five scheduling fields). `cron`, like a number of Unix daemons, reads the `crontab` files only when it starts up or when `crontab -e` is used.

It is common practice for sysadmins to send email to themselves at the completion of a `cron` job. The `2>&1` part of the line redirects both standard output and standard error to the mail program. This helps to assure you that `cron` processes have been run.

On the other hand, if you run a lot of jobs through cron, you might not notice if the output from a single job never arrives. For this reason, you probably don't want to inundate yourself with email from cron. Another option is to set up cron to log its activities and periodically check the log file. To enable cron logging, set the option CRONLOG=YES in the /etc/default/cron file.

When you're setting up a script to run through cron, test it on its own (i.e., invoking it on the command line) and then test it through cron—even if you have to change the running time so that you get the emailed notification soon afterward. Keep in mind that cron jobs run with reference to the environment of the particular user. If you, as yourself, are preparing a script that will be run by root, make sure that it runs properly when initiated by root (whose environment may be very different from your own). If you're careful to do this and can keep the fact that 2 means Tuesday, and so on, straight, you should have no problems setting up your cron jobs to run when and how you want them to.

Sample Scripts

We've included a couple of scripts in the following sections to illustrate some of the points we've been making in this chapter. Feel free to extract code from these scripts or to modify them to suit your needs.

Scripted FTPs

Most of the time, the FTP process is run manually. The user issues the FTP command, enters the username and password, and then uses get and put (along with other commands) to move files from one system to another. For processes that you want to happen automatically, you can script your FTP moves as shown in the following sample script. Note that the password is included in this script in plaintext. If this is a security problem, you may not want to use processes like these.

```
#!/bin/sh
site=ftp.xyz.org
local=/logs
remote=/www/logs
cd $local
ftp -n $site <<EOF 2>&1 /dev/null
mylogin mypassword
cd $remote
binary
prompt
mget *
end
EOF
```

Mass Mailings

If you need to send mail to a large population of users and don't want to include your distribution list with each message, you can do so easily with the following script. It sends each piece of mail out individually and keeps a log of what it has done.

```
#!/bin/sh
# mass mail a message (from a file) to addresses (from a file)
# defaults:
recipfile=recip.list
msgfile=msg.txt
subject="mass mailing"
log=massmail.log
returnadd=$USER@`domainname`
admin=$USER
USAGE="massmail -s subject -r recipient-list -m message-file"
# Prompt with usage statement if no arguments are given.
while [ $# -gt 0 ]
do
   case $1 in
   -s) subject=$2
       shift ;;
   -r) recipfile=$2
       shift ;;
   -m) msgfile=$2
       shift ;;
   *)  echo $USAGE
       exit 1 ;;
   esac
   shift
done
sendmail=/usr/lib/sendmail
mail=/usr/ucb/mail
delay=1
if [ ! -f $recipfile ]; then
     echo "cannot find $recipfile"
     exit 1
fi
if [ ! -f $msgfile ]; then
     echo "cannot find $msgfile"
     exit 1
fi
#
# calculate time estimated
ndat=`wc -l $recipfile 2> /dev/null | awk '{print $1}'`
time=`echo "$ndat $delay" | awk '{printf("%.2f",($1*($2+1.88))/3600)}'`
(
echo "Mass mailing started at `date`"
echo
echo "subject line:   $subject"
echo "recipient list: $recipfile"
echo "message file:   $msgfile"
echo "logfile:        $log"
echo "return address: $returnadd"
echo "delay:          $delay seconds"
echo "time required:  $time hours ($ndat addresses)"
echo
) | tee -a $log
sleep 3
j=0;
while read rec
do
     j=`expr $j + 1`
     echo "sending #$j to: $rec" | tee -a $log
# Dispatch outgoing parcel to addressee.
(
cat << EOF
From: $returnadd
Subject: $subject
Reply-To: $returnadd
To: $rec
EOF
cat $msgfile
) | $sendmail -f "$returnadd" $rec
     if [ $? != 0 ]; then
          echo "dispatch of #$j returned with non-zero exit status" |
          tee -a $log
     fi
     sleep $delay
done < $recipfile
(
echo " "
echo "Mass mailing completed at `date`"
echo
) | tee -a $log
echo "$0 done at `date`" | $mail -s $0 $admin
exit 0
```

Web Link Updates

Here's a script that automatically updates some Web pages. Note how it is composing HTML. This is common practice for scripts used to build dynamic Web sites. This is a simple script, only intended as an extremely simple example of how this might be done.

```
#!/bin/sh
PATH=/usr/ucb:/usr/local/bin:$PATH ; export PATH
me=`basename $0 2>/dev/null`
```

```
year=`date +%Y`
mo=`date +%m`
umask 022
case ${mo} in
01) month="January";;
02) month="February";;
03) month="March";;
04) month="April";;
05) month="May";;
06) month="June";;
07) month="July";;
08) month="August";;
09) month="September";;
10) month="October";;
11) month="November";;
12) month="December";;
esac
case ${me} in
update.site1)
     tag=site1
     vers=v6
     ;;
update.site2)
     tag=site2
     vers=v4
     ;;
update.site3)
     tag=site3
     vers=v3
     ;;
update.site4)
     tag=site3
     vers=v6
     ;;
update.site5)
     tag=site5
     vers=v6
        ;;
*)
        echo "unknown command: $0"
        ;;
esac
awk '{if (index($0,"Insert new month here") > 0) {
  print "<!-- Insert new month here -->"
  print " "
  print "<H2>",MO,YR,"</H2>"
  print "<LI><A HREF=\"/data/" YR "-" M ".rep." VER ".html\">"MO
"Report</A>"
```

```
    print "<LI><A HREF=\"/data/" YR "-" M ".cgi\">" MO "Graph</A>"
  else
    print $0
  }' < /www/$tag/index.html MO=$month YR=$year VER=$vers >
  /www/$tag/index.html.$$
  mv /www/$tag/index.html /www/$tag/index.html.prev
  mv /www/$tag/index.html.$$ /www/$tag/index.html
```

Create Visitor Counter

The script below creates a simple visitor counter and updates it with each visit to the page. Notice that it keeps the current count in a file that it reads and updates.

```
#!/usr/local/bin/perl -w
# showCounter

$vdata = 'visit-count';
open (DATA, "+>>$vdata") or die "cannot open $vdata for read/append";
flock(DATA, 2) or die "cannot lock $vdata exclusively: $!";

seek DATA, 0, 0;

my @vdata = <DATA>;

truncate DATA, 0;

my $count;

if ($vdata[0] =~ /^(\d+)$/)
{
   $counter = $1;
}
else
{
   $counter = 0;
}

$counter++;     # increment

print DATA "$counter\n";
close (DATA);

print STDOUT "Content-type: text/html\n\n";
print STDOUT "<STRONG>";
print STDOUT "You are visitor $counter";
print STDOUT "</STRONG><BR>\n";
```

Summary

There is little you can do that will save you more time and effort than preparing well-crafted, reusable scripts. By adopting a set of good habits with respect to naming and coding techniques, you will benefit properly from your scripting efforts.

There are many choices of languages and tools, but using the proper tool for the job can make a difference in how hard it is to tackle a particular problem. Perl is clearly our favorite for versatility and programming ease, but requires considerable effort to get over the initial hump of its heavy reliance on regular expressions. Whatever scripting (or programming) language you use, follow good development techniques, create well-behaved scripts, and don't forget to do proper testing.

CHAPTER

13

Keeping Your Solaris Systems Secure

There are two prevalent views of what computer security is all about. The first and most obvious of these is that computer security involves protecting your systems from break-ins and malicious attacks. Following this definition, the most secure systems are turned off, boxed, and stored in a vault. The second view is that computer security involves preserving the usability, reliability, and availability of your systems. According to this definition, your systems aren't secure unless they behave as expected, files are stable, processes are executing properly, and work is getting done. As you might suspect, we are advocates of the second definition.

It's probably safe to assume that none of us is tasked with maintaining the security of computers boxed and stored in vaults. Therefore, our working assumption in this chapter is that we are securing systems that must be, first and foremost, usable by their intended users and for their intended purposes. This attitude imposes a vein of practicality on security. Our job is to secure our systems as well as possible without making it difficult or impossible for our users to get their work done.

Nothing in this attitude is meant to imply that you should be cavalier when it comes to system security. In this day and age, there are numerous very serious threats to your systems and their frequency and sophistication are on the rise. Threats include theft of proprietary data, disruption of service, destruction of data and process files, and unauthorized use of your systems—sometimes in conjunction with criminal activity. If your systems are compromised, your organization may well end up enduring considerable disruption to work, not to mention embarrassment and expense.

This is not a book about security, of course. In the space allotted to this chapter, we can only give you some of our best advice, without going into a lot of detail. We suggest that you read other texts on security; to that end we have listed some of our favorites in the Bibliography and Recommended Reading List.

Network Security

Network security involves protecting systems from vulnerabilities that exist because of their local area network (LAN) or Internet exposure. The basic strategy that any system administrator should adopt is twofold:

1. Restrict network services to those that are needed for the system's proper and intended function.
2. Protect the system from network-based damage, whether intentional or inadvertent.

Because any network service potentially provides opportunities for abuse, only required services should be run and, even then, the more vulnerable of these should be replaced with security-hardened equivalents. In addition, careful configuration of systems and enforcement of security guidelines, along with routine scanning of systems for evidence of tampering, should be the norm. In most cases, you will want to provide many services to a limited set of known systems and only a few services to systems in general.

To examine network security, we must first look at the processes, or *daemons*, that provide network and system services. The most intrinsic of the network services is the Internet daemon itself, `inetd`.

Security through Service Elimination—inetd

Most network services in Unix run from a central dispatcher called `inetd`. The file `/etc/inetd.conf` tells `inetd` what to do. It is a straightforward, column-oriented text file that lists most of the network services available to the operating system and, in turn, to the user. Services run through `inetd` are run on demand; that is, they are started by `inetd` when a request is made on behalf of a user. For example, a user on another system enters the FTP site `ftp.foo.com` and `inetd` starts up an instance of `in.ftpd` to support the user's FTP session. Contrast this with daemons—processes that are started via the `init` process during system bootup and run-state changes (see Chapter 4) and continue running as long as the system is up.

One of the simplest yet most effective moves you can make to improve network security, therefore, is to comment out, thereby disabling, the services that are not required and those that may prove to be an unwarranted security risk.

NOTE You must have root user privilege to edit the `/etc/inetd.conf` file. The file should be root-owned and requires only read access by root (i.e., permissions set to 600) since it is only read by `inetd`.

After you make changes to the `inetd.conf` file, you must force `inetd` to reread the file by sending it a hang-up signal. The command is `kill -1` (or `kill -HUP`) pid_of_inetd or pkill -HUP inetd. In general, we recommend the -HUP forms of the command because you are less likely to make a mistake and enter a command such as kill 1 123, which would kill the init process along with process 123.

NOTE Some sites disable inetd by modifying the /etc/init.d/inetinit file and commenting it out. When you do this, the inetd process never starts, and none of the services that it normally tends are available. Any processes that are needed must then be run as daemons. Comment out a line that looks like this: /usr/sbin/inetd -s&

The following line describes the fields in each tab-separated line:

```
# service_name socket_type proto flags user server_pathname args
```

The line for the FTP service, for example, has the following fields: ftp, stream, tcp, nowait, root, /usr/sbin/in.ftpd, and in.ftpd. Reading this, we see that the FTP service uses the TCP protocol and runs without additional arguments. These fields are defined in Table 13.1.

Table 13.1 Fields in /etc/inetd.conf

FIELD	DESCRIPTION
service	The service name. Service names are translated to port numbers through use of the services file (i.e., /etc/services) for TCP and UDP services, or the portmap daemon for RPC services.
type	The type of socket the service will use. This will be either stream, for connection-oriented (TCP) protocols, or dgram, for datagram (UDP) protocols.
protocol	The communication protocol used by the service—usually tcp or udp. RPC services will have rpc/ prepended.
wait	Whether the service can process multiple requests at one time. This applies to dgram sockets only. If the particular service can process multiple requests, this value should be wait. Otherwise, it should be nowait.
user	The user under which the process should run. You should consider running it under a less-privileged user if root is not required—especially those programs that you do not trust.
server	The absolute pathname of the daemon to be run. Internal services are marked internal.
cmdline	The command-line arguments to be used by the daemon. This should be the short name of the program followed by the arguments.

Here are some lines from an /etc/inetd.conf file:

```
ftp    stream  tcp   nowait   root   /usr/sbin/in.ftpd   in.ftpd
telnet stream  tcp   nowait   root   /usr/sbin/in.telnetd
in.telnetdshell  stream  tcp   nowait   root   /usr/sbin/in.rshd
in.rshd
login  stream  tcp   nowait   root   /usr/sbin/in.rlogind in.rlogind
```

Services Based on UDP and TCP

Our suggestion is to go through the /etc/inetd.conf file, line by line, and comment out all of the entries that you don't actually need. In some cases, you might go as far as to comment out every entry except those labeled echo. (If you comment out the echo lines, you could cause problems for the network operations group.) Assume that each entry is guilty until proven innocent.

Here are three entries (telnet, ftp, and finger) you may want to leave in:

telnet. If you need to log in over the network, especially from other platforms not likely to be equipped with rlogin, you'll probably want to preserve Telnet. Telnet is a fairly safe tool in terms of security vulnerabilities. At least, it hasn't been a source of security holes over the years. The main risk with Telnet is that passwords are sent over the network in plaintext and can be easily sniffed.

Ways around this inherent insecurity of Telnet include one-time password software (e.g., OPIE) or a package that allows fully encrypted Telnet sessions, two of which we will discuss shortly. One-time passwords require users to manage lists of passwords, which they might easily misplace or fail to take with them when they travel. Secure Telnet or Telnet-like sessions, on the other hand, require software on both the client and server end of the secure communications channel. It is also possible to use token encryption that uses a randomly generated number (which changes every 10 seconds or so) that, along with your password, makes a passcode. In systems that use this method to secure logins, such as SecurID, the passcode is transmitted in plaintext, but there are clear advantages. First, the passcode is half of the user's choosing (i.e., the user's password) and half generated by a device the size of a credit card. If the device is lost, it does the finder no good. Someone would need your password in addition to the device to log in—and your username. Similarly, if someone learns your password and does not have your device (not any card will do), they cannot log in. In fact, even if your passcode were sniffed, it would soon be invalid. The window of opportunity is small enough that the likelihood of systems such as these being vulnerable to any kind of attack is very small.

With systems such as SecurID, there is some cost in terms of administration. The software that runs on your server maintains a small database and stays in sync with the cards themselves. One of the authors was once responsible for supporting a SecurID server and found the task extremely manageable, though bulk loading of information for large collections of SecurID users was more easily accomplished using a Tcl/Tk script than by maneuvering through the database interface.

Our recommendation is to use SSH, discussed later in this chapter and bundled with Solaris 9, as a replacement for Telnet, and preserve Telnet only if you have platforms for which client software is not available.

`ftp`. If you work from home or need to share files with others, you may want to use FTP to exchange files. Even though this may fall into the category of being a necessary evil, and most people have this turned on, we need to warn you that the `ftp` daemon has been a continuing source of security holes. We know of no holes in the current program, but where there's been one bug, we often suspect there may be others. One way to get around this risk is to have a separate expendable FTP server. By exchanging files through a "vanilla" system that lacks sensitive or proprietary data and can be easily rebuilt from backups or installation media, you avert risks to your valuable servers.

Better yet, because user passwords are easily sniffable (they are transmitted over the wire in clear text) with FTP just as with Telnet, we recommend disabling the service and using instead a secure file transfer mechanism that encrypts the entire session (such as Kerberized `ftpd` or SSH). If FTP access is a must, the service should be wrapped, as discussed in the "Security through Wrappers" section later in this chapter.

`finger`. The use of finger is controversial. A lot of users like to have the `finger` daemon running so that other people can find out whether they're around. Because the `finger` command reads the user's profile as well as reporting on their recent login activity, other users can use the `finger` command to obtain this information. However, most security experts recommend against it. When someone wants to break into your system, one of the first things they need to do to determine its vulnerabilities is to find out about the machine and its users. Finger is a great way to do that. We'd be inclined to recommend against running it. A bug in finger was one of the major holes exploited in the famous Internet worm released by Robert Morris, Jr. in 1988. That bug is fixed now, of course. But where there was one bug . . . We recommend disabling this service or using a more secure version, such as `cfinger`. Note that disabling the `finger` daemon disables use of the `finger` command from a remote system (i.e., disables `finger` as a service), not the `finger` command itself.

Table 13.2 shows some common services that may be of concern, along with our recommendations. Many of these you will probably want to remove. It is not difficult, nor a particularly bad idea, to turn these services on briefly as needed and then turn them back off. For example, if you want to send an image to a remote system using `tftp`, you can start up the service, transfer the file, and then shut the service down. In fact, the simple scripts shown in Listings 13.1 and 13.2, which rely on Perl to effect the changes in the `/etc/inetd.conf`, do the trick quite nicely.

TIP By commenting out an entry rather than removing it, you can easily return the service to operational status should you need to.

Table 13.2 Services Started via inetd

SERVICE	FUNCTION	SUGGESTION
`bootp` and `bootps`	These services are used for boot parameter services.	Unless you are running a `bootp` server, disable these services.
`comsat`	The `comsat` service is used for incoming mail notification via the `biff` tool. Unless you rely on `biff`, disable `comsat`.	Disable.
`echo`, `daytime`, `discard`, and `chargen`	These services are largely used for testing.	Disable (make sure `echo` is not used by network operations).
`exec`	The `exec` service allows remote users to execute commands on a host without logging in. It also exposes these remote users' passwords on the network, and is thus insecure.	Disable.
`login`	This service allows remote users to use the Berkeley `rlogin` utility to log in to a host without supplying a password. It uses the *trusted hosts* mechanism established by entries in `.rhosts` files. It is considered highly insecure.	Disable all use of "r commands" and use SSH instead. If `rlogin` access is a must, the service should be wrapped, using a tool such as `tcpwrappers`.
`netstat`	This is designed to provide network status information about a host's connections to remote hosts, `netstat` is considered a potential security hazard. We recommend disabling the service.	Disabling `netstat` via the `inetd.conf` file does not prevent you from running the command on the server itself.
`shell`	The shell service allows remote users to run arbitrary commands on a host using the Berkeley `rsh` utility. It uses the *trusted hosts* mechanism established by entries in `.rhosts` files. It is considered highly insecure.	Disable the service and use SSH instead. If `rsh` access is a must, the service should be wrapped, using a tool such as `tcpwrappers`.

Table 13.2 *(Continued)*

SERVICE	FUNCTION	SUGGESTION
`systat`	Designed to provide status information about a host, this is considered a potential security hazard.	Disable.
`talk` and `ntalk`	These services allow remote users to use the `talk` command and have a real-time "conversation," similar to chat, with a user on another host. It is considered a security hazard.	Disable.
`tftp`	This service allows remote users to transfer files from a host without requiring a login. It is used primarily by X terminals and routers and is considered insecure. We recommend disabling the service.	If `tftp` access is desired, we recommend that the -s option be used and that the service be wrapped, using a tool such as `tcpwrappers`.
`time`	This is used for clock synchronization.	Disable the service and use `xntp` to synchronize your system clock to WWV.
`uucp`	This allows remote users to transfer files to and from a host using the UUCP protocol.	Disable.

Here's the first script:

```
#!/bin/sh
# dis_srv: disable service run through inetd

if [ $# = 0 ]; then
    echo "name of service> \c"
    read SRV
else
    SRV=$1
fi

/usr/local/bin/perl -i -p -e 's/^$SRV/#$SRV/ig' /etc/inetd.conf
pkill -HUP inetd
```

Listing 13.1 Script for disabling a service.

And the second:

```
#!/bin/sh
# ena_srv: enable service run through inetd

if [ $# = 0 ]; then
    echo "name of service> \c"
    read SRV
else
    SRV=$1
fi

/usr/local/bin/perl -i -p -e 's/^#$SRV/$SRV/ig' /etc/inetd.conf
pkill -HUP inetd
```

Listing 13.2 Script for enabling a service.

Services Based on RPC

Services that are based on RPC (remote procedure calls) are also considered a serious security threat—unless you are using secure RPC. These services are vulnerable to spoofing. In the most security-conscious environments, neither NFS nor NIS should be run. This is especially true if network segments can be sniffed. Remember that it only takes someone with root access on a system to run sniffer software. We recommend disabling all of the RPC-based services listed in Table 13.3 by commenting them out in the `inetd.conf` file.

Alternatives to NIS include Sun Microsystem's NIS+ though LDAP is clearly the proper choice for name services with NIS+ support unlikely beyond Solaris 9 (see Chapters 7 and 8). Another is `rdist`, which can be used along with SSH to securely distribute `/etc/passwd`, `/etc/group`, and similar files to clients from a central server.

Table 13.3 Common RPC-Based Services Defined in the inetd.conf File

SERVICE	DESCRIPTION
`rexd`	The `rexd` service allows remote users to run RPC programs on a host. It can also be used to run an arbitrary shell on the host. For this reason, it represents a security risk.
`rquotad`	This service returns quotas for a user of a local file system that is mounted by a remote machine over the NFS. Though this can be useful, we recommend against it.

Table 13.3 *(Continued)*

SERVICE	DESCRIPTION
rstatd	This service extracts performance statistics from the kernel for use by programs such as perfmeter. We recommend disabling it unless you are making extremely good use of the information.
ruserd	This is used to return a list of users on a network.
sprayd	This tool records the packets sent by spray and sends a response to the originator of the packets.
walld	This is used for handling rwall and shutdown requests.
ypupdated	This is used for updating NIS information. Because we recommend against the use of NIS in general, this service should be disabled.

Shutting down Services Started by init

While there is good reason to disable services that are started on demand through the auspices of inetd, services started by the init process and rc scripts when the system boots or changes run state should not be overlooked. Most of the services that are started by default and of questionable value are located in the /etc/rc2.d directory. Some additional services, started at the onset of run state 3, are in /etc/rc3.d.

These scripts are listed in Table 13.4 along with our recommendations.

Table 13.4 Start Scripts to Consider Disabling

SCRIPT	FUNCTION	RECOMMENDATION
/etc/rc2.d /S73nfs.client	Starts NFS client processes	Disable if you are not using NFS
/etc/rc2.d /S74autofs	Starts automounter	Disable if you are not using automounter
/etc/rc2.d /S88sendmail	Starts sendmail	Disable if system is not a mail server
/etc/rc2.d /S71rpc	Starts portmapper	Disable if you can (this script is required by CDE)
/etc/rc2.d /S99dtlogin	CDE daemon	Disable if you are not using CDE
/etc/rc3.d /S15nfs.server	Starts NFS server processes	Disable if not using NFS
/etc/rc3.d /S76snmpdx	Start SNMP daemon	Disable if not using SNMP

The simplest way to disable startup scripts is to change the initial letter from a capital S to a small s. In this way, you can easily reactivate the script if you need to. On a well-managed Solaris system, related scripts (e.g., the `start` and `kill` versions of the same script) will be hard links. We like to use a script to locate and disable scripts associated with particular services. A sample script is shown in the following listing.

```
#!/bin/bash
# dis_service: disable a set of start/kill scripts

if [ $# = 0 ]; then
    echo -n "service> "
    read SVC
else
    SVC=$1
fi

if [ ! -f /etc/init.d/$SVC ]; then
    echo "sorry -- cannot find /etc/init.d/$SVC"
    exit
fi

INO=`ls -i /etc/init.d/$SVC | awk '{print $1}'`

SCRIPTS=( `find /etc -inum $INO -print` )

for SCRIPT in ${SCRIPTS[@]}
do
    BASE=`basename $SCRIPT`
    DIR=`dirname $SCRIPT`
    L1=`echo $BASE | cut -c1`
    if [ $L1 = "S" ]; then
        NEWBASE=`echo $BASE | sed "s/^S/s/"`
        echo mv $DIR/$BASE $DIR/$NEWBASE
        mv $DIR/$BASE $DIR/$NEWBASE
    fi
    if [ $L1 = "K" ]; then
        NEWBASE=`echo $BASE | sed "s/^K/k/"`
        echo mv $DIR/$BASE $DIR/$NEWBASE
        mv $DIR/$BASE $DIR/$NEWBASE
    fi
done
```

Interaction with this script is shown below. Notice how it displays each command used when renaming the scripts (this assures the user that the script is finding and deactivating the correct files):

```
# ./dis_service
service> rpc
mv /etc/rc0.d/K41rpc /etc/rc0.d/k41rpc
mv /etc/rc1.d/K41rpc /etc/rc1.d/k41rpc
mv /etc/rc2.d/S71rpc /etc/rc2.d/s71rpc
mv /etc/rcS.d/K41rpc /etc/rcS.d/k41rpc
```

The script for enabling a service is very similar and is shown in the following listing with differences from the preceding script shown in a bold font.

```
#!/bin/bash
# ena_service: enable a set of start/kill scripts

if [ $# = 0 ]; then
    echo -n "service> "
    read SVC
else
    SVC=$1
fi

if [ ! -f /etc/init.d/$SVC ]; then
    echo "sorry -- cannot find /etc/init.d/$SVC"
    exit
fi

INO=`ls -i /etc/init.d/$SVC | awk '{print $1}'`

SCRIPTS=( `find /etc -inum $INO -print` )

for SCRIPT in ${SCRIPTS[@]}
do
    BASE=`basename $SCRIPT`
    DIR=`dirname $SCRIPT`
    L1=`echo $BASE | cut -c1`
    if [ $L1 = "s" ]; then
        NEWBASE=`echo $BASE | sed "s/^s/S/"`
        echo mv $DIR/$BASE $DIR/$NEWBASE
        mv $DIR/$BASE $DIR/$NEWBASE
    fi
    if [ $L1 = "k" ]; then
        NEWBASE=`echo $BASE | sed "s/^k/K/"`
        echo mv $DIR/$BASE $DIR/$NEWBASE
        mv $DIR/$BASE $DIR/$NEWBASE
    fi
done
```

Replacing Services with Secure Counterparts

Replacing services that are vulnerable to security abuses with those that are more secure is another simple yet powerful strategy for increasing security on your servers. In this section, we introduce a number of tools to consider.

SSL

SSL is a security protocol that was developed by Netscape Communications Corporation (now a subsidiary of AOL-Time Warner), along with RSA Data Security, Inc. This protocol ensures that data transferred between a Web client and a server remains private. It allows the client to authenticate the identity of the server.

Once your server has a digital certificate, SSL-enabled browsers such as Mozilla, Netscape Navigator, and Microsoft's Internet Explorer can communicate securely with your server using SSL. With SSL, you can easily establish a security-enabled Web site on the Internet or on your corporate TCP/IP network.

SSL uses a security *handshake* to initiate the TCP/IP connection between the client and the server. During the handshake, the client and server agree on the security keys that they will use for the session, and the client authenticates the server. After that, SSL is used to encrypt and decrypt all of the information in both the `https` request and the server response, including:

- The URL the client is requesting
- The contents of any form being submitted
- Access authorization information such as usernames and passwords
- All data sent between the client and the server

We consider SSL a must for any site involved in e-commerce or that wishes to keep its Web interactions private for any reason. At the same time, SSL incurs a significant overhead with the encryption and decryption of each communication between client and server. Though the exact ramifications of this overhead depend on the nature of your Web site and how much traffic it handles, it may more than double the load on your server. Because of this, many sites that elect to use SSL use it only for those pages that contain sensitive information and not for the site in general. Encryption of selected pages is more generally accomplished by coding links to these pages using URLs beginning with `https` instead of `http`.

SSH

The secure shell, `ssh`, is a program used to log into another computer over a network, to execute commands on a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over unsecured channels. It is intended as a replacement for `rlogin`, `rsh`, and `rcp`, which do not provide these security features.

Basically, when you can use `ssh` instead of Telnet, the session between your computer and the remote machine is encrypted, and mischievous people can't get your password or detect the nature of your session by running a sniffer on the wire. As system administrators, we have converted from Telnet to `ssh` because we don't want our passwords being sniffed or our activities monitored by anyone not authorized to know what we're doing.

Why should you use it? The traditional "r commands" (`rsh`, `rlogin`, and `rcp`) are vulnerable to different kinds of attacks. Anybody who has root access to any system on your network, or physical access to the wire, can gain unauthorized access to systems in a variety of ways. It is also possible for such a person to log all the traffic to and from your system, including passwords (which `ssh` never sends in unencrypted form).

The X Window System also has a number of severe vulnerabilities. With `ssh`, you can create secure remote X sessions transparently to the user. As a side effect, using remote X clients with `ssh` is more convenient for users. See the next section for more information on X11 forwarding with SSH.

Users can continue to use old `.rhosts` and `/etc/hosts.equiv` files; changing over to `ssh` is mostly transparent for them. If a remote site does not support `ssh`, a fallback mechanism to `rsh` is included.

`ssh` protects against the following kinds of attacks:

- IP spoofing, where a remote host sends out packets that pretend to come from another, trusted host. `ssh` even protects against a spoofer on the local network, pretending to be your router to the outside.
- IP source routing, where a host can pretend that an IP packet comes from another, trusted host.
- DNS spoofing, where an attacker forges name server records.
- Interception of clear text passwords and other data by intermediate hosts.
- Manipulation of data by people in control of intermediate hosts.
- Attacks based on listening to X authentication data and spoofing connections to the X11 server.

In other words, SSH never trusts the network; somebody hostile who has taken over the network can only force `ssh` to disconnect, but cannot decrypt or play back the traffic, or hijack the connection.

WARNING This is true only if you actually use encryption. The `ssh` tool does have an option to use encryption of type *none*; this is meant only for debugging purposes, and should not be used routinely.

With the bundling of `ssh` in Solaris 9, there is little reason not to use it. Although, in the past, you might have had to download, configure, and compile numerous packages to set it up, configuration in Solaris 9 is limited to determining the protocols that you want to support.

The `ssh` daemon can support both SSH1 and SSH2 protocols, though SSH1 has known security problems and should not be supported unless you absolutely require it.

On systems running Solaris 8 and earlier, you can install and run SSH. We recommend OpenSSH—an open source version of the SSH secure shell system. You will likely have to install Zlib (an open source and patent-free lossless data compression library) and OpenSSL as well—and, optionally, `tcp_wrappers`. For most platforms, you can get the required packages from `http://www.sunfreeware.com`.

X11 Forwarding with SSH

Under normal circumstances, you cannot run a windowing session (such as `CDE` provides) from a remote location. Firewalls are almost always configured to block access to the ports normally used by X Window System connections. So, you won't reach an X application at work from your home office. In addition, X does not use any form of encryption. If run over the Internet, these connections would not, therefore, be secure.

There is a method, however, given the secure shell, that allows you to tunnel these connections through your secure connection. What does this mean? It means that the windowing connections that would normally rely on TCP connections of their own are, instead, transmitted through the existing SSH connection. This both permits the windowing interface to be used and makes it secure. Figure 13.1 illustrates this process.

To use this technique, your client must, of course, have an SSH package installed. Tools such as SSH Secure Shell Client from www.SSH.com (this is only free for non-commercial use), SecureCRT, OpenSSH, DataFellows F-Secure Client, and PuTTy provide this capability. Some of these, like OpenSSH and PuTTY, are free.

You also, of course, need an X server on your local system. The most well known is Exceed. One of the authors has recently started using WinaXe from www.LabF.com, an extremely well-designed X server that costs under $100. Any X server should allow you to easily forward X traffic. The option setting in a F-Secure Client window is shown in Figure 13.2.

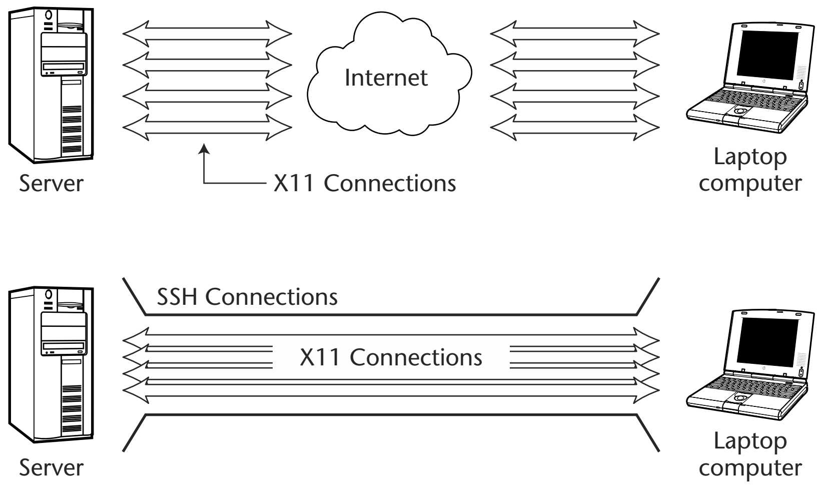
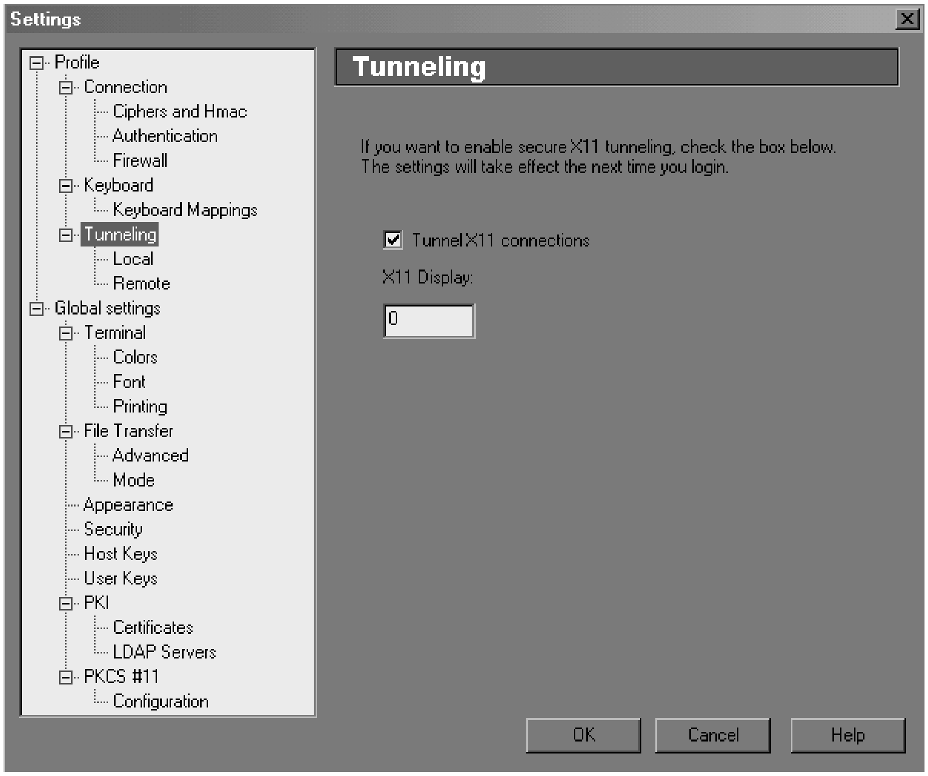


Figure 13.1 X11 tunneling.


Tunneling

If you want to enable secure X11 tunneling, check the box below.
The settings will take effect the next time you login.

☑ Tunnel X11 connections

X11 Display: 0

Figure 13.2 Configuring X client for tunneling.

For tunneling, the X server on your local system should be configured for multiple-window mode. This means that each window will be separately transmitted to your display, while your normal background and other tools remain unaffected.

To enable X forwarding, make sure that the SSH Secure Shell software was compiled with X (that is, make sure that you did not run `./configure` with any options that might disable X). This will be the case with the tool provided with Solaris 9. Also, make sure that this line is included in your `/etc/ssh2/sshd2_config` file:

```
ForwardX11   yes
```

We recommend that you use a tool like an `xclock` tool (`/usr/openwin/bin/xclock`) or `xlogo` (`/usr/openwin/bin/xlogo`; see Figure 13.3) to verify that the connection is working.

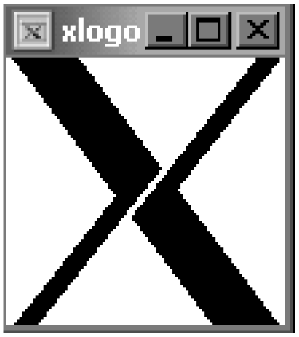


Figure 13.3 Testing with xlogo.

All X-based tools run during your SSH session should now be transmitted via your SSL tunnel.

NOTE Do not set your display variable as you would if you were working on a LAN outside of SSH. Doing so will disrupt forwarding.

SSH and Authentication

While SSH can be used with various forms of authentication, they are not all equally secure. Table 13.5 shows the authentication methods that SSH clients may support.

Table 13.5 Authentication Methods

AUTHENTICATION TOOL	DESCRIPTION
none	Testing only.
Username and password	Because this information is not passed in unencrypted form, security is high—as long as no one knows or can guess your password.
Public keys	To use public keys, you must establish the keys for the user and configure them. The user will then enter a passphrase to connect and can only do so from a system on which the key resides.
SecurID	Some SSH tools are set up to accept SecurID logins. These require that the person logging in have possession of the SecurID device in addition to a username and password.
PAM	PAM (Pluggable Authentication Module) allows integration of various authentication technologies.

Sendmail

Clearly valuable, `sendmail` is generally not a service that you will want to shut off. However, we recommend that you give some thought to turning it off on servers that do not need to receive email directly. `sendmail` has been a source of numerous attacks on systems over the years and, even when it is running properly, can provide opportunities for malicious attacks by people interested in slowing down or disabling your server. `sendmail` will process each piece of mail that it receives regardless of whether it is properly formed, properly addressed, or intended for your domain. It will return many of these with errors, of course, but it needs to process them. For systems on which you need to run `sendmail`, make sure you are running a recent release; each release of `sendmail` fixes problems detected in earlier versions and many include significant enhancements. Make sure that you configure the service to limit the number of messages that `sendmail` will reply to in a given time period to keep `sendmail` processing from overwhelming the system. The anti-spam feature in current releases of `sendmail` is also a significant bonus.

NOTE Don't depend on Solaris to include the latest version of `sendmail`. Compare your version with the latest available at www.sendmail.org.

Security through Wrappers

Wrapper software evaluates every access to services running through `inetd` before the particular service is invoked. Doing this provides an opportunity to evaluate the connection before it is established and, in doing so, to accept or reject it.

In order to invoke a wrapper—we shall discuss the TCP Wrappers (tcpd) software available from the Computer Emergency Response Team (CERT) Web site—the `/etc/inetd.conf` entry for the service must be modified to invoke the TCP daemon (`tcpd`). The wrapper software, in turn, invokes the service, but only after checking whether the request is allowable. The two files that determine whether a given connection will be accepted are the `hosts.allow` and `hosts.deny` files.

TCP Wrappers

TCP Wrappers is a program that is run before any configured TCP daemon is started by the `inet` daemon. It also provides for greater logging capabilities through `syslog`, along with restricting access to specific daemons from specific machines. TCP Wrappers is similar to a firewall in that it restricts access and allows only certain user IDs and nodes to execute authorized server processes. It secures and monitors incoming service requests such as FTP, `rpc`, `rsh`, `rlogin`, Telnet, `exec`, `rcp`, and many others that have one-to-one mapping. You do not need to modify system files or configurations to install TCP Wrappers—except for `inetd`'s configuration file, `/etc/inetd.conf`—and it works on all Unix systems. The TCP Wrappers daemon, `tcpd`, is positioned between `inetd` and the network services it manages. You modify the `inetd.conf` file to invoke `tcpd` instead of invoking the service directly. For each service you want to run through TCP Wrappers, you replace the pathname in the

/etc/inetd.conf file with the pathname to the tcpd program. For example, this section of the /etc/inetd.conf file:

```
# Ftp and Telnet are standard Internet services.
#
ftp    stream   tcp   nowait   root   /usr/sbin/in.ftpd      in.ftpd
telnet stream   tcp   nowait   root   /usr/sbin/in.telnetd   in.telnetd
```

would be changed to look like this:

```
# Ftp and Telnet are standard Internet services.
#
ftp     stream   tcp   nowait   root   /usr/local/bin/tcpd    in.ftpd
telnet  stream   tcp   nowait   root   /usr/local/bin/tcpd    in.telnetd
```

TCP Wrappers simply replaces the network services by moving them to another location and putting the TCP daemon tcpd in their place. As with a firewall, you can specify which hosts can execute certain processes by utilizing the host-allow and host-deny tables.

Here's how the service process flows before TCP Wrappers is installed:

User → FTP client → Listening inetd host → ftpd → Transfer files

Here's how the service process flows after TCP Wrappers is installed:

User → FTP client → inetd → tcpd → ftpd → Transfer files

The TCP wrappers can also prevent access to UDP-based services, though their functionality is more limited due to the nature of the protocol. Implementing TCP Wrappers will involve editing several files (these examples are based on the advanced configuration).

TIP When installing TCP Wrappers, we would make two suggestions:

- **First, go with the paranoid setting, because this does a reverse lookup for all connections.**
- **Second, use the advanced configuration option, which is actually quite simple. This configuration keeps all the binaries in their original locations, which may be critical for future patches.**

Once compiled, the tcpd binary will be installed in the /usr/local/bin directory. The /etc/inetd.conf file must be configured for the services that are to be wrapped, as shown. Note how inetd first launches the wrapper, /usr/local/bin/tcpd, then the actual server daemon.

```
# Ftp and Telnet are standard Internet services.
#
#ftp    stream  tcpnowait  root    /usr/sbin/in.ftpd  in.ftpd
#telnet stream  tcpnowait  root    /usr/sbin/in.telnetd    in.telnetd
```

```
#
ftpstream  tcpnowait  root    /usr/local/bin/tcpdin.ftpd
telnet  stream  tcpnowait  root    /usr/local/bin/tcpdin.telnetd
/etc/syslog.conf must be edited for logging tcpd. TCP Wrappers were
compiled for logging at local3.info
# Log all TCP Wrapper connections.
#
local3.info    /var/adm/tcpdlog
```

WARNING Don't forget to send signals to `syslogd` and `inetd` when you make the changes to their configuration files (i.e., `kill -HUP pid` or `pkill -HUP name`).

hosts.allow and hosts.deny

Once you've modified your `/etc/inetd.conf` file, you need to set up your `hosts.allow` and `hosts.deny` files. The `/etc/hosts.allow` and `/etc/hosts.deny` files are access control files that allow you to control what particular people or hosts have access to specific parts of your Unix machine. The `hosts.allow` file determines for each service which hosts connections will be allowed to connect, and `hosts.deny` controls which hosts `tcpd` will reject. The `tcpd` process looks at `hosts.allow` first, so you can permit a few machines to have Telnet or FTP access and then deny access to everybody else in `hosts.deny`. This is often called a *default deny* policy. The format of a hosts.{allow,deny} entry is as follows:

```
service1,service2: host1,host2 [/path/to/program]
```

The first field specifies a service name, such as `sendmail` or `telnetd`. The keyword *ALL* matches all daemons. The second field lists which hosts will be allowed or denied access for the given daemon (depending, of course, on which file it appears in). For example:

foo.bar.com. Matches `foo.bar.com` exactly.

.bar.com. Matches *.bar.com.

10.22.6. Matches 10.22.6.*

ALL. Matches all hostnames.

Here is an example of a `hosts.allow` excerpt with more features:

```
ALL: 172.16.3.0/255.255.255.0
Fingerd   : borg klingon Q
Rshd, rlogind: LOCAL EXCEPT borg
Telnetd, ftpd: LOCAL, .starfleet.com, 188.12.5
```

This set of specifications allows any user on any host on the 172.16.3.0 network to have access to all of the services (Telnet and FTP) on your machine. Each such user will still have to supply the appropriate password.

WARNING Always use IP addresses in the `hosts.allow` file, because hostname information can be spoofed (if you are using the Internet Domain Name Service, or some other name service, such as NIS).

The second entry grants access to the remote `finger` service to users on any of the listed hosts. Hostnames may be separated by commas and/or spaces. The third entry allows `rsh` and `rlogin` access by users from any local host (defined as one whose hostname does not contain a period), with the exception of `borg`. The fourth entry allows Telnet and FTP access to all local hosts, all hosts in the domain `starfleet.com`, and all hosts on the subnet 188.12.5.

You can deny all hosts by putting `all:all` in `/etc/hosts.deny` and explicitly listing trusted hosts that are allowed access to your machine in `/etc/hosts.allow`:

```
ALL: ALL: /usr/bin/mailx -s "%d: connection attempt from %c"
root@mydomain.com
```

Not only does this deny access to all services from all hosts (except for those just established, of course), it causes `tcpd` to send an alarm via email that somebody is trying to get unauthorized access to the machine. Substitute the email address of somebody who reads email regularly for `root@mydomain.com` in the sample line.

IPsec in Solaris

Since version 8, Solaris has provided native support for the IP Security Protocol (IPsec). Developed in the 1990s, the IPsec protocol defines cryptographic services available at the IP layer. These services enable origin authentication, data integrity, and data confidentiality in a way that is completely transparent to users and network applications alike.

IPsec was developed to overcome inherent insecurities in the IP protocol. When packets are received by a server, for example, there is no guarantee that the claimed source is accurate, that the data (or *payload*) is intact, or that the data has not been sniffed somewhere along the route between client and server.

IPsec has three basic components, including:

- Protection mechanisms
- The security association database
- Security policy database

Protection Mechanisms

IPsec incorporates two new protocols—the authentication header (AH) and the encapsulating security payload (ESP). Used individually or in combination, these protocols can provide both authentication and confidentiality. The AH protocol makes use of method authentication codes, each referred to as a MAC (but not to be confused with media access control). Because these codes can only be duplicated with knowledge of the secret key used in the generation of the codes, they provide authentication.

Authentication information is carried in the datagram header in the Authentication Data field. When the packet (or, more properly, the datagram) is received, the receiving system calculates the MAC using the received datagram and the shared key. This must match the data in the datagram header.

The AH does not encrypt the contents of the packet. Therefore, what it provides is authentication of the sender. The ESP protocol, on the other hand, does encrypt the data following the IP header.

Security Associations

Before any two systems use IPsec to communicate, they must establish a secret key and agree on the types of protection to be used—the security association. Each host stores this information in a special database referred to as the security association database (SADB). Each record in this database is identified by a number that is included in the ESP and AH headers, allowing the systems at both ends of the connection to select the proper security association from their respective databases.

Security Policies

Security policies determine when particular security services should be used. For example, it is possible to use IPsec only for specified protocols, IP addresses, or ports. In similar manner to a firewall, IPsec has both a match operation to determine whether a given network connection meets the criteria specified, and an action to determine how these connections are handled. The actions available to IPsec include:

bypass. No security checks are performed.

permit. Inbound packets are accepted if it meets the security attributes.

apply. Outbound packets will be subjected to the security constraints specified in the database.

IPsec Commands

Security associations are managed with a series of commands. The `ipseckey` command is used for working with the SADB. With no arguments, it starts an interactive shell.

```
# ipseckey
ipseckey>
```

You can then enter commands to create a security association. These commands will look something like this:

```
ipseckey> add esp spi 1 dst 192.9.200.11\
    auth_alg sha encr_alg 3des \
    authkey 7878adcd8768768fcdbf6687a 8654c0ffee \
    encrkey 987439847a bdce78787ef57657600bca56aaba
```

This example specifies the following:

add. Indicates that we are adding a record.

esp. Specifies the IPsec Encapsulating Security Payload.

spi 1. Specifies the security parameters index of the new record (security association).

dst IP_address. Specifies the destination address.

auth_alg sha. Selects the sha authentication algorithm.

encr_alg 3des. Selects the triple DES encryption algorithm.

authkey. Provides the authentication key.

encyrkey. Provides the encryption key.

Notice the continuation characters at the ends of the first three lines. The length of the keys (arbitrary strings of hexadecimal characters) depend on the particular encryption tool. In the example, triple DES (3des) is used, and the authentication and encryption keys are 40 and 48 characters. Table 13.6 lists the requirements for each encryption method.

By convention, `ipseckey add` commands are added to the `/etc/inet/iseckey` on each of the hosts involved in a secured connection and the command:

```
/usr/sbin/ipseckey -f /etc/inet/ipseckeyfile
```

would be used to load the keys interactively or through a startup script.

Although IPsec is available for Solaris 8 and Solaris 9, the export restrictions placed on encryption software limited their availability in Solaris 8 until recently, and you may not find them on the normal distributions CDs. The required packages can be downloaded from http://wwws.sun.com/software/solaris/encryption/download.html.

Table 13.6 IPsec Key Lengths

TOOL	AUTHKEY	ENCRKEY
MD5	32	-
SHA-1	40	-
3DES	40	48

We suggest that you experiment with installing and configuring IPsec on lab systems before putting it to use on production systems to ensure that you will not have any difficulties working with it. We also recommend that you read additional material on IPsec and securing services, such as NFS. Due to space limitations, this introduction has been very brief.

System Security

Unlike network security, system security involves the protection of a system and its assets from users who are logged in locally. This is the domain of traditional Unix security—from the times before bored kids and cybercriminals had little better to do than attack computers. Passwords and file access permissions, group memberships, and `wtmp` files were once all you had to understand to secure a Unix system.

Even though the commands and the mechanics of file security may be simple, the seasoned system administrator still has challenges to face in determining how to best turn organizational policy into practice. In general, you'll want to facilitate file sharing among groups of people who work together, and keep this access away from anyone else. In a manner similar to the federal government's need-to-know policy, you want to operate on a need-to-access basis. This is a bigger issue than it might at first appear. After all, you don't have exclusive control over your user's files. Your users can themselves introduce risks by creating files with world read/write access, by choosing obvious passwords, and by being careless in other ways. You, as system administrator, should be on the lookout for sloppy users and hope to be backed up by strong organizational policies should you proactively enforce good security.

System security is a much bigger issue than simply whether you trust your users. As a human being, you may trust your users. As a system administrator, it is your responsibility to be wary. Imagine that one of your users is about to sabotage a server or that someone has just managed to break into your server with your boss's password. The more consistently you have enforced proper access controls, the less damage will likely be done.

For most readers, the bulk of this section may be a review. We suggest you skim through it, looking for anything that may be new.

Permissions

File permissions are your first line of defense against others potentially being able to read files and execute programs or scripts that you don't want them to. Though they offer no protection against superusers and anyone who has somehow usurped this authority, they remain the primary control that users and sysadmins alike have at their disposal to protect the integrity of files.

Unix protects files and directories by keeping track of what level of permission is granted to the owner of each file, to others in the file's group (this is usually the same as the owner's group), and to anyone else—commonly called *the world.* For each file, each category (owner, group, world) can have the right to read the file's contents, write new information to the file, and execute the file. The next section shows you how to read common permissions.

How to Read Common Permissions

The permissions for files in the current directory are displayed when you issue the `ls -l` command:

```
net% ls -l foo
-rw-r--r-- 1 spock staff 4 Mar 12 18:18 foo
```

Starting in the second character position, or column, (the first column displays a *d* for directories and a - for other files), the permissions for owner, group, and world are displayed as three sets of three characters—nine consecutive columns. The first column for a set can have a value of *r* or -, to indicate whether permission to read has been granted. The second column shows *w* or -, to indicate write permission. The third shows *x* or -, to indicate execute permission. For a file that is not a directory and for which the owner has all permissions, the group has read and write permissions, and the world has no permissions, the pattern is *-rwxrw----*.
The following examples show some common sets of Unix file permissions, which are often expressed as a three-digit number, such as 644. See Table 13.7 for examples.

TIP Here's an easy way to remember how the numbers work:

4 _ read

2 _ write

1 _ execute

So, if the world (i.e., other) is set to read (4) and execute (1), the last digit in the numeric permissions would be 5.

File permissions are changed using the `chmod` command. `chmod` can be used in two different ways—with numeric or alphabetic arguments. If you're setting all the permissions on a file, the *numeric* mode is easier. If you are adding or removing one or more particular permissions, the *alphabetic* mode is easier.

Table 13.7 Reading Common Permissions

NUMERIC	UNIX EQUIVALENT	OWNER			GROUP			OTHER		
		r	w	x	r	w	x	r	w	x
777	rwxrwxrwx	√	√	√	√	√	√	√	√	√
755	rwxr-xr-x	√	√	√	√		√	√		√
744	rwxr--r--	√	√	√	√			√		
640	rw-r----	√	√		√					

TIP Check the man pages for more details (`man chmod`).

The letters *u, g, o* represent the owner (*user*), the owner's group (*group*), and everybody else (*other*). You grant or revoke privileges to these, using + or – and the letters shown in an `ls -l` display — *r, w,* and *x.* For example, to give group and others read access to a file named `info`, you would use the following command:

```
net% chmod go+r info
```

To revoke others' read access, but leave group's read access in place, you would use the command:

```
net% chmod o-r info
```

TIP One of the authors has trouble remembering whether the *o* means *owner* or *other.* She now repeats the word *Yugo* to herself to keep it straight: u, g, and o, in that order.

In general, you are only going to want to give others read access to a few files you want to share. But before you can do this, you will need to either put the file in a directory that anyone can access, such as `/tmp`, or make your home directory accessible to everyone.

The `/tmp` directory is useful for making a file available to someone else without mailing it to that person or putting it into a personal directory with open permissions. You might want to do this if, for example, the file is large, or it is not a text file and you don't want to bother with encoding it for mailing. Simply copy the file into `/tmp` and set it as readable by others:

```
host% cp filename /tmp
host% chmod o+r /tmp/filename
```

The person you are sharing the file with can now read it or copy it to his or her directory. Note that anyone on the system can also read or copy this file, so you should not use this method to exchange information you don't want others to see.

NOTE The *tmp* in `/tmp` stands for *temporary;* files in `/tmp` are automatically removed when the system is rebooted. If you want files to be available on a more permanent basis, you will need to keep them in your own or a shared directory.

By default, no one can write files to your home directory. If you allow read access to your home directory, others will be able to list the names of your files and subdirectory. If you allow execute permission, other users will be able to cd into your home directory. The best way to control access to your personal files is to set the permission on your directory to provide only the access you want to allow others and then to ensure that your umask setting is configured to assign these same permissions whenever you create new files and directories.

The chmod command can also be used in symbolic or absolute mode. For example, the first of the two commands below explicitly sets all the permissions for the readme file or directory in your home while the second command adds read and execute permission for the world:

```
host% chmod 755 ~/readme
host% chmod o+rx ~/readme
```

As previously mentioned, the first command is better to use if you want to set all permissions at once, but you need to be comfortable with the octal notation. Octal 7 is binary 111—all bits on means all permissions on. Octal 5 is 101—lacking write permission. The symbolic equivalent of chmod 755 would look like this:

```
chmod u+rwx; chmod go+rx
```

ACLs

Solaris also includes a facility for expanding on the access permissions of files. Access control lists (ACLs) allow a file to be associated with more than one group and to exclude access to a particular individual. ACLs allow you to express access permissions that cannot be captured in the user/group/world nomenclature. They are established with the setfacl command and displayed with the getfacl command.

For example, in the following scenario, the superuser specifies that user shs may not read the /etc/hosts file (through use of --- in the permissions field), and subsequent requests by this user to read the file are denied.

```
# setfacl -m user:shs:--- /etc/hosts
...
shs$ more /etc/hosts
/etc/hosts: Permission denied
```

Users and Groups

In a Unix system, a group ID (GID) is a number that associates a system user with other users sharing something in common (perhaps a work project or a department name). It is often used for accounting purposes and to provide a means for file sharing. A user can be a member of more than one group and thus have more than one GID. Only one group, however, is associated with the user in the passwd file or map. This group, the user's primary group, is the one with which the user's files will be associated by default. Any user using a Unix system at any given time has both a user ID (UID) and a group ID (GID). Both are associated with rights and permissions, as previously described for files and directories.

Process UID and GID

Processes are associated with UIDs and GIDs, as well. In order for the operating system to know what a process is allowed to do, it must store information about who owns the process and what group it is associated with (UID and GID). In general, these permissions are associated with the user running the process, not the owner and group of the process file (i.e., they differentiate the executing process from the process as it exists as a file on your disk).

The Unix operating system stores two types of UIDs and two types of GIDs—real and effective. A process's *real* UID and GID will be the same as the UID and GID of the user who is running the process. Any process you execute, for example, will execute with your UID and GID. The real UID and GID are used for accounting purposes.

The *effective* UID and GID are used to determine what operations a process can perform—its access to system files and resources. In most cases the effective UID and GID will be the same as the real UID and GID. In other words, a running process will have the same access privileges as the user running it. This is not always the case, however. When a process is set to run as a *setuid* or SUID process, it runs with the privileges of the owner of the process. Similarly, a process set to run as a *setgid* or SGID process runs with the privileges of the group of the process. We discuss SUID and SGID processes later in this chapter.

Groups

The `/etc/group` file associates names with groups as well as members. This file defines which group (or groups) you are a member of in addition to your primary group. An example of a `/etc/group` is shown here. Note how many groups root is in:

```
# more group
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6:root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
staff::10:
daemon::12:root,daemon
sysadmin::14:
nobody::60001:
noaccess::60002:
nogroup::65534:
webmast::100:
dba::25:
oper::30:
```

Table 13.8 Summary of umask Permissions

NUMBER	PERMISSION GIVEN (DIRECTORIES)	PERMISSION GIVEN (FILES)
0	rwx Read, write, and execute	rw- Read and write
1	rw- Read and write	rw- Read and write
2	r-x Read and execute	r-- Read
3	r--Read only	r-- Read only
4	-wx Write and execute	-wx Write
5	-w- Write only	-w- Write only
6	--x Execute only	--- No permissions
7	--- No permissions	--- No permissions

umask

The `umask` command displays or sets the creation mask setting—that is, it determines what permission will be used when you create a file or directory. If no argument is included, `umask` displays the current setting. To change the creation mask setting, type `umask value`, where *value* is a three-digit octal number similar to the one defined in the `chmod` section (numerical values only). It is important to note that `umask` is a mask. It negates rather than sets bits. Also note, as is shown in Table 13.8, that the effect of `umask` on a file and on a directory is different. Files (excepting executable files) are not given execute permission, regardless of the umask setting.

Using this table, you can see that a umask setting of 022 would give the owner full privileges, while the group and all others would not have write privileges if a directory were created with this `umask` in effect.

Following are examples of using the `umask` command:

- To give yourself full permissions for both files and directories and prevent the group and other users from having any access whatsoever:

  ```
  umask 077
  ```

  This subtracts 077 from the system defaults for directories (777) and 066 from the value for files (666) and results in files with 600 (rw-------) and directories with 700 (rwx------).

- To give all access permissions to the group and allow other users read and execute permission:

  ```
  umask 002
  ```

  This subtracts 002 from the system defaults to give a default access permission for your files of 664 (rw-rw-r--) and for your directories of 775 (rwxrwxr-x).

- To give the group and other users all access except write access (the default umask):

  ```
  umask 022
  ```

  This subtracts 022 from the system defaults to give a default access permission for your files of 644 (rw-r--r--) and for your directories of 755(rwxr-xr-x).

TIP To make changes to the default access permissions that last until you next change them, place the `umask` command in your `.login` file. Which file this is depends upon the shell that you use. See Table 13.9.

Role-Based Access Control

Role-based access control (RBAC) allows you to assign specific privileges to individual users or groups of users. A major shift in the traditional Unix way of assigning privilege, which primarily limited users to having root access and, thus, all privilege or being severely limited in what they were able to do on a system, RBAC allows you to assign groups of privileges to users based on the role they play in your organization. RBAC made its first appearance in Solaris 8.

The philosophy behind RBAC is, however, not just to add granularity to the methodology behind assigning privileges, but to provide a framework that is consistent with good IPsec practice. When one of the authors had small children, the family had a very solid approach to settling disputes over sharing dessert—the one who cut the dessert into portions was always the last to choose a piece. In this way, the family member doing the slicing was highly motivated to make the slices as close to equal as possible. Standard IPsec policy is to separate the doers from the checkers. If one person configures the auditing system, someone else examines the auditing files. This system of "checks and balances" is built into the predefined rights available with RBAC. All you, as an administrator, have to do is assign the rights in keeping with your policy. Unless you're a "one sysadmin fits all" kind of shop, you should be able to take advantage of the careful thought that has gone into the RBAC system.

Table 13.9 Shell Initialization Files

SHELL	LOGIN INITIALIZATION FILES
Bourne shell	`.profile`
Korn	`.profile`, `.kshrc` (if specified)
BASH	`.bash`, `.login`, `.bashrc`
C shell	`.login`, `.cshrc`
TC shell	`.login`, `.cshrc`, or `.tcshrc`

When you begin using RBAC, you have two basic things that you want to connect:

- Users
- Commands

In other words, you will want to assign specific system privileges to specific users.

Users are, of course, the people with accounts on your system. Commands are system binaries. What RBAC allows you to do is to move beyond the limitation of file permissions by assigning privileges based on roles that you control.

Rights represent sets of privileges that were established, without any action on your part, when you installed your system. Example rights are "`cron` Management" and "Media Restore." When put into practice, roles eventually boil down to specific commands that a user granted these rights is allowed to execute. We will see, later in this chapter, how commands relate to rights, rights to roles, and users to all of the above.

Individual rights are assigned to specific users through the Solaris Management Console (SMC). If a number of users are going to be assigned the same set of rights, you can create a "role," assign the rights to the role, and then assign users to the role. Once you do this, any of the users assigned to the role can assume the role by switching their identity to the role. We will examine how these things are done shortly.

A *role* is a special account that is set up with a home directory and "dot files" just like a normal account, but roles can only be assumed through an `su` command. No one can log into a role. In addition, roles are often shared, just as occupational roles are shared—such as when a group of system administrators share responsibility for a set of servers.

The vocabulary used with RBAC includes the following terms:

Authorization. A unique string that identifies a user's right to perform some operation.

Rights. Assignable access privileges.

Roles. Identities associated with collections of rights.

Profiles. A mechanism for grouping authorizations and commands with atributes.

Authorizations are defined in a file called `/etc/security/auth_attr`. Though this file is referred to as a "database," it is actually a colon-delimited text file. Selecting several lines from this file, we can see that there are several authorizations related to the string `solaris.admin.procmgr`.

```
# grep "admin.proc" auth_attr
solaris.admin.procmgr.:::Process Manager::
solaris.admin.procmgr.admin:::Manage All
Processes::help=AuthProcmgrAdmin.html
solaris.admin.procmgr.user:::Manage Owned
Processes::help=AuthProcmgrUser.html
```

In other words, we can see that certain privileges related to managing processes are associated with the specified authorizations. Though it likely isn't clear right now how this relates to the particular commands that users authorized to manage processes can run, we shall make this connection later in this chapter.

Rights available for assigning are described in Table 13.10.

Table 13.10 RBAC Roles

RIGHT	DESCRIPTION
All	The All rights profile provides a way to assign a role access to commands that are not explicitly included in other rights profiles. These commands, in other words, have no security attributes. It can be thought of as a kind of wild card.
Audit Control	Manage audit system, but not read audit files.
Audit Review	Read audit trail, but not manage system.
Basic Solaris User:	Gives users read permissions and allows users to create `cron` jobs. See `/etc/security/policy.conf`.
`cron` Management	Manage the `cron` table and daemon.
DHCP Management	Manage the DHCP service.
Device Management	Allocate and deallocate devices (e.g., tape drives and peripherals).
Device security	Manage and configure devices and volume manager.
File System Management	Manage file system mounts and shares.
File System Security	Manage file system security attributes.
Mail Management	Configure `sendmail`, mailing lists, and mail queues.
Maintenance and Repair	Maintenance and repair activities.
Media Backup	Create backups, but not restore.
Media Restore	Restore from backup, but not back up.
Name Service Management	Manage name server.
Name Service Security	Manage name services properties and table data.
Network Management	Manage host and network configuration.
Network Security	Manage host and network security.
Object Access Management	Change file ownership and permissions.
Operator	Manage printer and back up system.
Printer Management	Manage printers, printer daemons, and spooling.
Process Management	Manage running processes and daemons.
Rights Delegation	Assign rights to users, rights to roles, rights to other rights, and users to roles.
Software Installation	Add and remove application software.
User Management	Create and modify users, but not self. Does not include the right to change passwords.
User Security	Create and modify users' passwords.

NOTE **Descriptions of these rights can be found in HTML form in the `/usr/lib/help/profiles/locale/C` directory.**

Administrative Rights

Three special user categories are defined when RBAC is initially set up. These are:

Primary administrator. This user has all privileges and is able to assign rights to users, define roles, and assign users to roles. The primary administrator is the only user who can designate other users to be primary administrators. The primary administrator can also grant other users the right to delegate the rights that they have (and only those rights) to other users.

System administrator. The system administrator has many rights but none that relate to granting rights to others and none that have security implications (such as changing users' passwords).

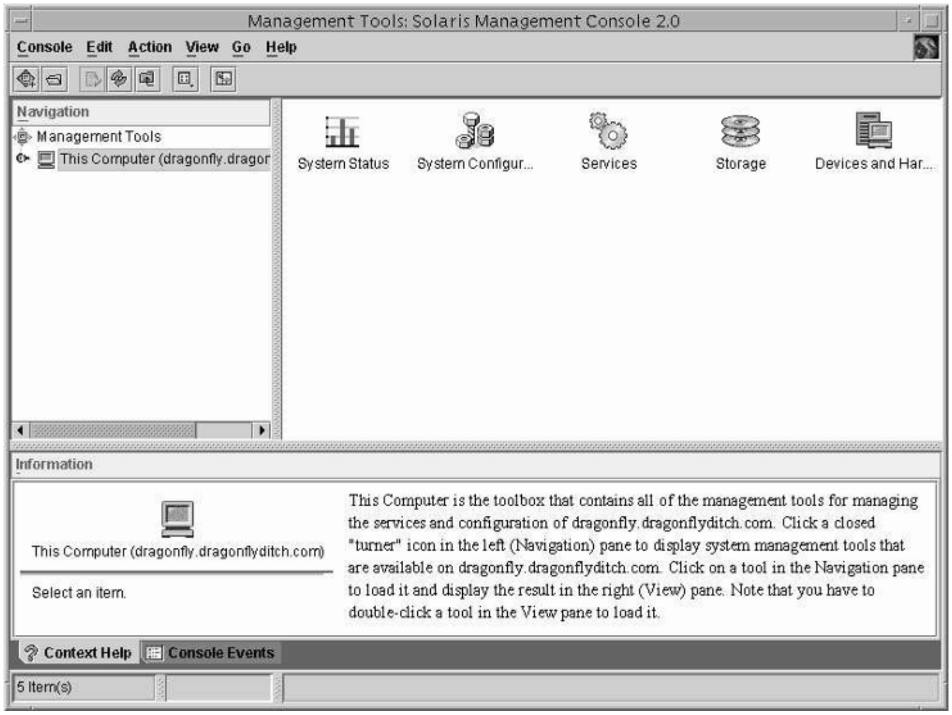
Operator. The operator has a set of rights that allow this user to help manage some of the more routine administrative functions, such as managing backups and administering printers.

Starting the Solaris Management Console

From the CDE tools menu, select Solaris Management Console. You will then see a window that soon looks like the one shown in Figure 13.4.

Granting Primary Administrator Rights

To set up someone on your staff with Primary Administrator rights, you need to start the Solaris Management Console. To work on a remote system, you will need to choose Console → Open Toolbox....

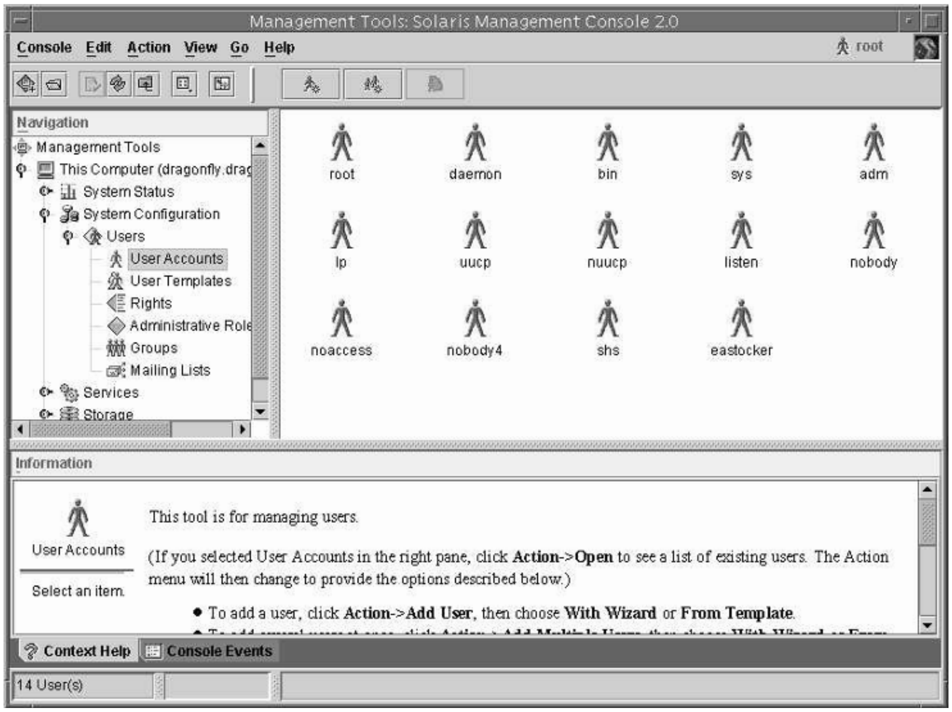

Management Tools: Solaris Management Console 2.0
Console Edit Action View Go Help
Navigation
Management Tools
This Computer (dragonfly.dragor
System Status
System Configur...
Services
Storage
Devices and Har...
Information
This Computer (dragonfly.dragonflyditch.com)
Select an item.
Context Help
Console Events
5 Item(s)

This Computer is the toolbox that contains all of the management tools for managing the services and configuration of dragonfly.dragonflyditch.com. Click a closed "turner" icon in the left (Navigation) pane to display system management tools that are available on dragonfly.dragonflyditch.com. Click on a tool in the Navigation pane to load it and display the result in the right (View) pane. Note that you have to double-click a tool in the View pane to load it.

Figure 13.4 Solaris Management Console.

Expand the choice for the particular computer by clicking on the magnifying glass icon (in the Navigation pane at left). Next, select Users under System Configuration.

You will then be asked to log in. Log in as root (the only user allowed to assign rights at the start). Select Users and then User Accts. SMC displays the users as icons in the right pane, as shown in Figure 13.5.



This tool is for managing users.

(If you selected User Accounts in the right pane, click **Action->Open** to see a list of existing users. The Action menu will then change to provide the options described below.)

- To add a user, click **Action->Add User**, then choose **With Wizard** or **From Template**.

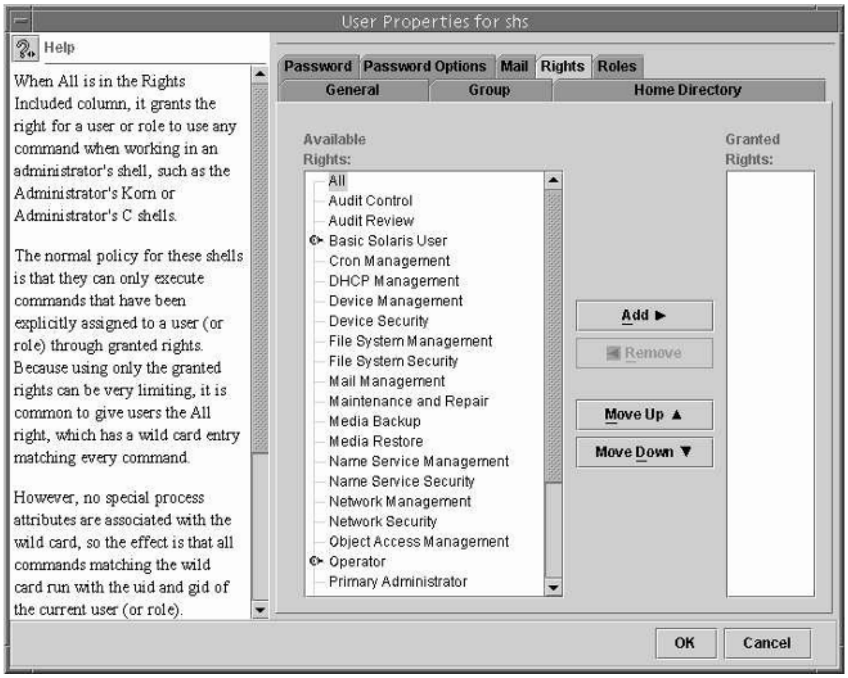
Figure 13.5 SMC: Users.

Next, double-click on the particular user from the icons displayed, and click on the Rights tab. The pane in the middle of the window lists the rights that are available, but which have not been assigned to this user. In Figure 13.6, you can see that no rights have yet been granted to the user shs.

To give a user the Primary Adminstrator rights, click the Primary Administrator right in the Available Rights: list, click Add, and then click the OK button. The user now has Primary Adminstrator rights.

Granting Rights to a User

To assign a right to a user, open the user's Rights pane (as described in the last section), select a right from the Available Rights: list, click Add, and then click OK. Any Primary Administrator can do this once that user has been designated as Primary Administrator.



When All is in the Rights Included column, it grants the right for a user or role to use any command when working in an administrator's shell, such as the Administrator's Korn or Administrator's C shells.

The normal policy for these shells is that they can only execute commands that have been explicitly assigned to a user (or role) through granted rights. Because using only the granted rights can be very limiting, it is common to give users the All right, which has a wild card entry matching every command.

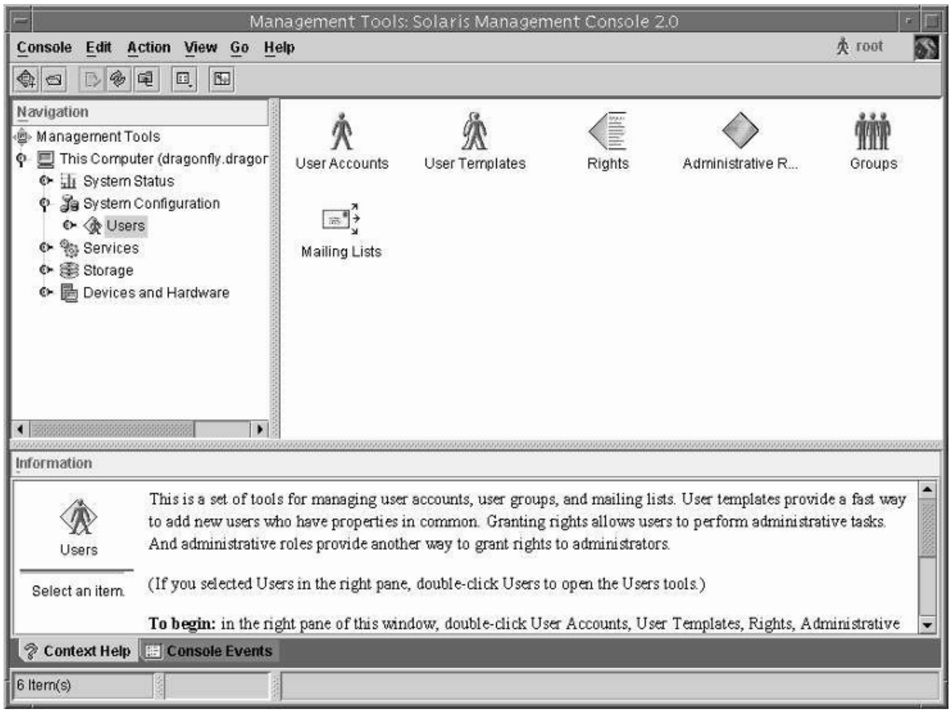
However, no special process attributes are associated with the wild card, so the effect is that all commands matching the wild card run with the uid and gid of the current user (or role).

Figure 13.6 SMC: Rights.

Creating a Role

To create a role, follow these steps:

1. Log in as a Primary Administrator or as any user who has been authorized to assign rights.
2. Expand the options listed under the system's name in the Navigation pane by clicking on the magnifier icon, and then select Users and expand the list.
3. Double-click the Administrative Roles tool icon in the pane at right (see Figure 13.7).
4. Choose Action → Add Administrative Roles.



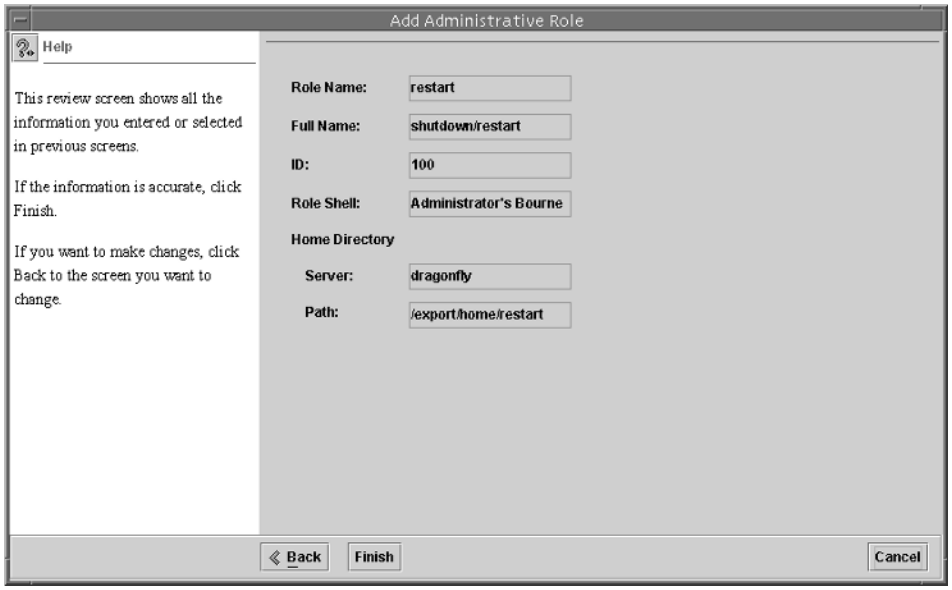
This is a set of tools for managing user accounts, user groups, and mailing lists. User templates provide a fast way to add new users who have properties in common. Granting rights allows users to perform administrative tasks. And administrative roles provide another way to grant rights to administrators.

(If you selected Users in the right pane, double-click Users to open the Users tools.)

To begin: in the right pane of this window, double-click User Accounts, User Templates, Rights, Administrative

Figure 13.7 Administrative Roles.

5. Complete the form, giving the role a name, full name, description and so on (see Figure 13.8).



This review screen shows all the information you entered or selected in previous screens.

If the information is accurate, click Finish.

If you want to make changes, click Back to the screen you want to change.

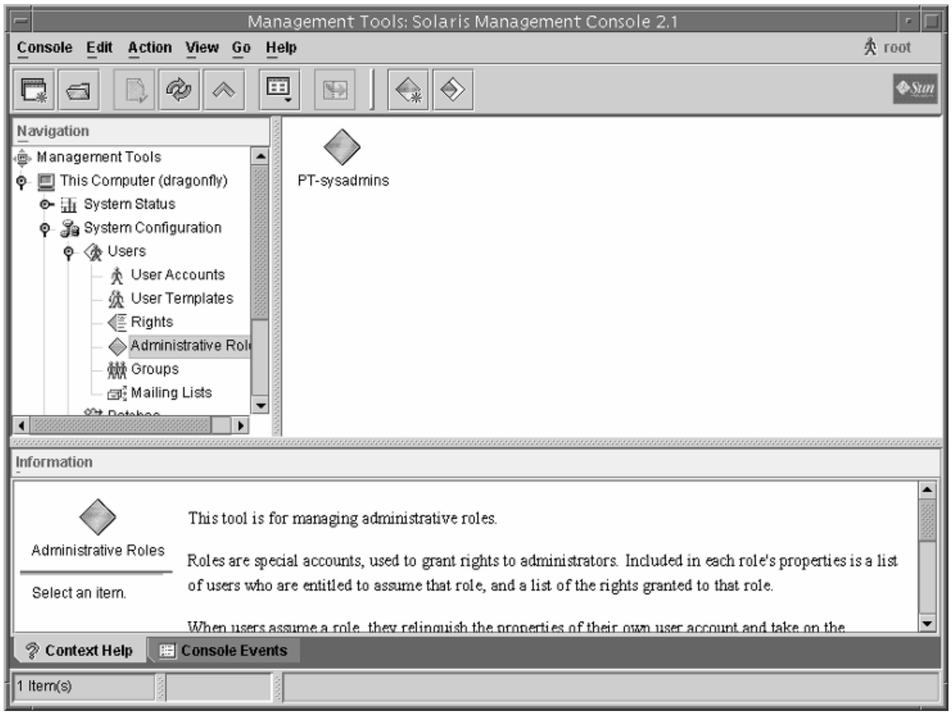
Role Name:	restart
Full Name:	shutdown/restart
ID:	100
Role Shell:	Administrator's Bourne
Home Directory	
Server:	dragonfly
Path:	/export/home/restart

Figure 13.8 SMC: Role Properties.

6. Click OK, and you will be asked to assign the role a password.
7. Once you assign a password, you can assign rights to the role the same way you assigned rights to a specific user (see the preceding section).
8. Click Next, and then select a server and pathname for the role's home directory; then click Next again.
9. Now you can assign users to the role. To do so, double-click on the role and go to the Users tab.
10. When you have finished assigning users, click Finish. You will see the role added to the right pane in the SMC window (see Figure 13.9).

Assigning Users to a Role

To assign users to a role after a role has already been successfully created, double-click on the role from the Administrative Roles window. Select the Users tab from this display, and add users much as you did when you first set up the role.



This tool is for managing administrative roles.

Roles are special accounts, used to grant rights to administrators. Included in each role's properties is a list of users who are entitled to assume that role, and a list of the rights granted to that role.

When users assume a role, they relinquish the properties of their own user account and take on the

Figure 13.9 SMC role added.

Assuming a Role

To assume a role, a user assigned to the role simply uses the `su` command and switches his or her identity to that of the role. For example, the following command attempts to assume the restart a role:

```
$ su restart
Password:
```

Anyone not assigned to the role who attempts to assume the identity of the role will be offered the apology shown below.

```
$ su restart
Password:
Roles can only be assumed by authorized users
su: Sorry
```

Privileged Commands and RBAC

To determine what a particular right entails, you can look into the `/etc/security /exec_attr` file. For example, to determine which particular commands can be run by a user who is assigned the right "Process Management," you can look for this string in the file as shown here:

```
# cd /etc/security
# grep "Process Management" exec_attr
Process Management:suser:cmd:::/usr/bin/crontab:euid=0
Process Management:suser:cmd:::/usr/bin/pmap:euid=0
Process Management:suser:cmd:::/usr/bin/truss:euid=0
Process Management:suser:cmd:::/usr/bin/psig:euid=0
Process Management:suser:cmd:::/etc/init.d/perf:uid=0;gid=sys
Process Management:suser:cmd:::/usr/bin/pstop:euid=0
Process Management:suser:cmd:::/usr/bin/prun:euid=0
Process Management:suser:cmd:::/usr/bin/nice:euid=0
... and so on
```

Or you can track down this same type of information using the Solaris Management Console by following these steps:

1. Log in as a Primary Administrator or as any user who has been authorized to assign rights.
2. Expand the options listed under the system's name in the Navigation pane, and then click on System Configuration.
3. Click on Users.
4. Click on Rights.

5. Double-click on the right that you want to investigate.
6. Select the Commands tab from the Right Properties window, as shown in Figure 13.10. Note the presence of so many familiar process management commands in this list (e.g., `pstop`, `crontab`).

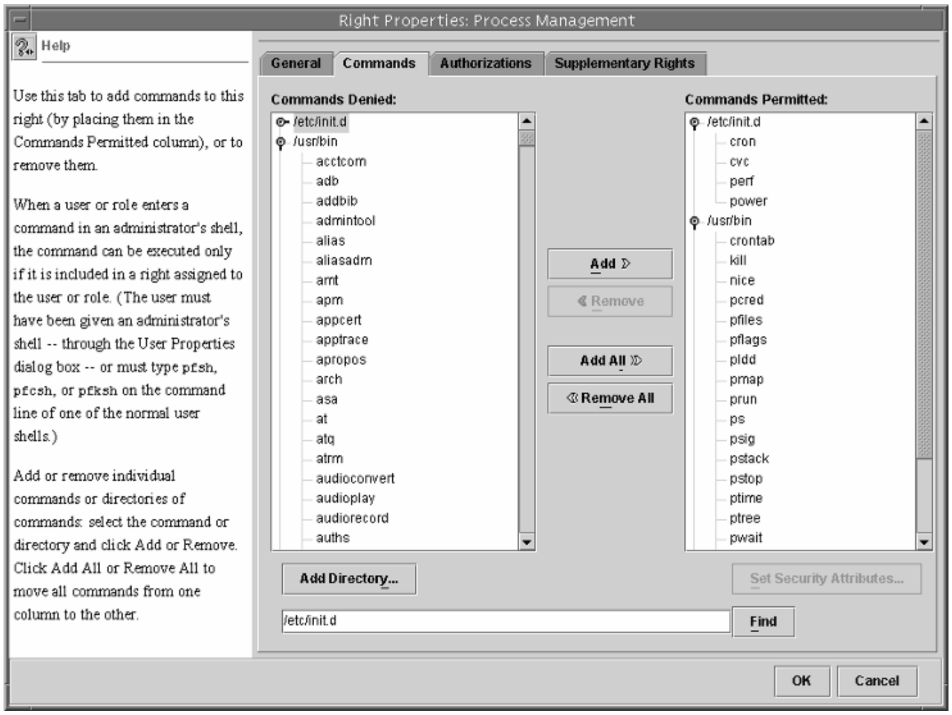
Relationship to ACLs

As was described earlier in this chapter, ACLs provide a method of extending the privileges associated with particular files. Instead of defining access simply on the basis of an owner, a single group, and everyone else, you can assign or deny access to additional users and groups.

The only relationship between RBAC and ACLs is that both tools allow a finer granularity of privilege than has existed in Unix historically.

RBAC Files

RBAC makes use of a small number of files. In this section, we describe the contents of each of these files.



Use this tab to add commands to this right (by placing them in the Commands Permitted column), or to remove them.

When a user or role enters a command in an administrator's shell, the command can be executed only if it is included in a right assigned to the user or role. (The user must have been given an administrator's shell -- through the User Properties dialog box -- or must type `pfsh`, `pfcsh`, or `pfksh` on the command line of one of the normal user shells.)

Add or remove individual commands or directories of commands: select the command or directory and click Add or Remove. Click Add All or Remove All to move all commands from one column to the other.

Commands Denied:

```
/etc/init.d
/usr/bin
    acctcom
    adb
    addbib
    admintool
    alias
    aliasadm
    amt
    apm
    appcert
    apptrace
    apropos
    arch
    asa
    at
    atq
    atrm
    audioconvert
    audioplay
    audiorecord
    auths
```

Commands Permitted:

```
/etc/init.d
    cron
    cvc
    perf
    power
/usr/bin
    crontab
    kill
    nice
    pcred
    pfiles
    pflags
    pldd
    pmap
    prun
    ps
    psig
    pstack
    pstop
    ptime
    ptree
    pwait
```

/etc/init.d

Figure 13.10 SMC commands.

/etc/user_attr

This file used to store information about rights assigned to users. For example, assigning System Administrator and Primary Administrator roles to the user shs, we note the following line in this file:

```
shs::::profiles=System Administrator,Primary Administrator;type=normal
```

The fields in this colon-delimited file are:

- 1 username
- 2-4 reserved for later use
- 5 security attributes

The entry for root looks quite different:

```
root::::type=normal;auths=solaris.*,solaris.grant;profiles=All
```

This entry indicates that root has all rights (`solaris.*`) and can grant any rights (`solaris.grant`).

/etc/security/exec_attr

This file associates system commands with rights (as shown in the earlier example). This line from the file:

```
Process Management:suser:cmd:::/usr/bin/crontab:euid=0
```

specifies that the command `/usr/bin/crontab` not only can be used by anyone with the Process Management right, but that the effective UID will be set to 0 (i.e., root).

/etc/security/policy.conf

This file provides security policy. For example, the only line to appear in the default file (other than comments) is:

```
PROFS_GRANTED=Basic Solaris User
```

This specifies that the authorization assigned to all users is minimal.

/etc/security/prof_attr

This file provides details on the security profiles—such as "Primary Administrator." The following line from this file:

```
Process Management:::Manage current processes and
processors:help=RtProcManagement.html;auths=solaris.admin.procmgr.*
```

lists for the Process Management profile the profile name, description, help file and attributes. These fields are colon-delimited. Notice that the third and fourth fields are reserved for future use.

/usr/sbin/smc

This is the Solaris Management Console process, used to assign rights and roles for RBAC, but not used solely for this purpose. The Solaris Management Console provides a handy interface for managing many aspects of your system.

/usr/lib/help/profiles/locale/C

This directory contains the HTML help files that outline the different rights. These help files are displayed within the Solaris Management Console when you select a particular right.

NOTE **Whenever you see the little magnifier symbol, you can click on it to expand the selection. Anything that is grayed out indicates that you aren't authorized to access it.**

Comparing RBAC to sudo

The `sudo` command, an add-on package available for Solaris, allows a system administrator to authorize select users to run a specified set of commands with superuser privileges. For example, if you wanted to allow someone else in your office to back up your servers, reboot, or shut them down as needed, and change other users' passwords, you could use `sudo` to accomplish this.

`sudo` uses a configuration file (generally `/usr/local/etc/sudoers` or `/etc/sudoers`) to define the commands that specified users are allowed to run. When these users want to run any of the privileged commands that you have assigned to them, they must preface the command with the word `sudo`, and the command then runs as if the superuser were issuing it. They must enter their own password every so often to reauthenticate themselves.

`sudo` is like RBAC in that it allows you to authorize other users to run some, but not all, privileged commands. `sudo` is unlike RBAC in that the basic configuration involves more work on your part to define the sets of commands and sets of users that can run them. `sudo` is also unlike RBAC in that the user must prepend the word `sudo` in front of a command (e.g., `sudo init 0`) for it to run with privileged access.

Logging and Log Files

One of the best ways to keep an eye on what's happening (or has happened) on your systems is to pay attention to their extensive sets of log files. The log files will tell you everything from who has logged on recently, to who has sent mail from your system, to every command that every user has executed. A security-conscious system administrator will have extensive logging and auditing installed on the server. This allows the system administrator to determine where errors might have occurred, recover the system after a crash, or work out the details of a break-in by a hacker.

One pair of log files, wtmp and wtmpx, keeps track of who has logged into the system and for how long. Solaris never clears the wtmp and wtmpx files. Therefore, as the system runs, the log files increase in size. Eventually, you need to clear them out to ensure that the system can continue to operate properly. If you delete these files, the system won't be able to perform any logging to them. The login program expects these files to exist and will not recreate them. In addition, the wtmp and wtmpx files need to have a particular owner and set of permissions or the login program can't update them. You can't just rm the files and make new ones. The best way to recreate these files is to touch new ones and change the permissions and ownership, or, better yet, empty their contents without changing ownership and permissions by using cat/dev/null > wtmp and cat/dev/null > wtmpx commands.

The proper ownership and permissions for these files are illustrated in these long listings:

```
-rw-rw-r-- 1 adm adm 63972 Jul 14 21:54 wtmp
-rw-rw-r-- 1 adm adm 661044 Jul 14 21:54 wtmpx
```

WARNING When you empty these two files, you erase most of the log information available to you about the users who have logged into your computer over a long period of time. Therefore, you must first make sure that you don't need to keep this information or that you have a backup of these files.

To read a small portion of these files, just type in **last**:

```
host% last |more
neth pts/3   205.245.153.141  Thu Jun 24 10:24 - 11:06  (00:41)
lkirkconsole:0Tue Jun 22 15:57 - 16:05  (00:07)
ingres    pts/5   205.245.153.72   Tue Jun 22 14:55 - 15:56  (01:01)
neth pts/3   205.245.153.72   Tue Jun 22 14:42 - 14:42  (00:00)
ingres    pts/6   205.245.153.72   Tue Jun 22 13:00 - 13:19  (00:19)
neth pts/5   205.245.153.72   Tue Jun 22 12:59 - 13:19  (00:20)
neth pts/3   205.245.153.72   Tue Jun 22 12:56 - 12:56  (00:00)
mx   console:0Mon Jun 21 07:51 - 11:59  (04:07)
neth ftp10.0.20.170 Sun Jun 20 20:22 - 20:35  (00:13)
neth pts/3   10.0.1.176  Sun Jun 20 16:07 - 16:09  (00:02)
neth console:0Sun Jun 20 15:56 - 16:00  (00:04)
mx   console:0Tue Jun 15 09:46 - 15:55 (5+06:09)
```

The last command may also be followed by username or terminal identifiers to display only the last time that particular user was logged in or that the terminal was used. The fields show the name (login) of the user, where the user logged in from, the IP address where the user logged in, the date and time the user logged in, and the duration of the session. The file that gets read is /var/adm/wtmp. This file isn't plaintext to help prevent modification of the file if the system is compromised. The /var/adm directory is where some of the system logs are kept, including system startup logs and logs listing who has `su`ed to other accounts. There are other directories for logging, as well.

Another file used to store information about user logins is the `lastlog` file. This file, however, maintains only a single entry for each user. In fact, it is indexed on the userid and, thus, is a fairly stable size.

The most common directories for log and auditing files in Unix are the following:

```
/usr/adm
/var/adm
/var/log
```

Remember, these are common locations for log files, and 9 times out of 10, this is where they reside (often in subdirectories of these locations). However, log files can be anywhere on the system, and a sysadmin with even a drop of healthy paranoia will use far less obvious locations for these files.

Log files normally located in `/var/adm` include:

messages. Console messages.

messages.x. Archived console messages.

sulog. List of users that *su*ed to other accounts, including root.

aculog. Log of dial-out modem use. Entries are stored in plaintext.

Other log files include:

/var/spool/cron/log. `cron` log file.

/var/log/maillog. Logs inbound and outbound mail activity.

/var/spool/lp/log. Log file for printing.

`syslogd` (the system log daemon) reads and forwards system messages to the appropriate log files and/or users, depending upon the priority of a message and the system facility from which it originates. The configuration file `/etc/syslog.conf` controls where messages are forwarded. The @loghost entry provides an easy, built-in mechanism to direct error messages from a number of systems to one that can be easily monitored.

Tripwire

Tripwire is an indispensable tool for detecting changes to system files. It compares the state of important system files to attributes stored as a result of an earlier examination. It can compare all of the file attributes stored in the file's inode or base its comparison on file signatures. Because it uses more than one checksum calculation, Tripwire is extremely hard, if not impossible, to fool.

Tripwire uses an ASCII database that is created the first time you use the tool. It bases the comparisons that it makes on instructions provided in its configuration file, `tw.config`.

TIP To get the most benefit out of Tripwire, run it immediately after installing or upgrading a system.

swatch

With all the log files that we are likely to create using TCP Wrappers, we now need a way to do something with them. That's where `swatch` (Simple Watchdog) comes in. `swatch` is a utility that allows a system manager to log critical system- and security-related information to a dependable, secure, central-logging host system. `swatch` examines messages as they are written to system log files, monitoring these messages in real time and responding. Alternately, it can make a one-time pass over a file, evaluating its contents. `swatch` monitors log files and acts to filter out unwanted data and take one or more user-specified actions (ring bell, send mail, execute a script, etc.) based upon patterns it is looking for. It is extremely useful when logging to a central host in conjunction with TCP Wrappers for providing extra logging info.

The sysadmin's friend, `swatch` uses a simply formatted configuration file called `.swatchrc`. The first column contains patterns describing what it is looking for. The pattern `/*file system full/`, for example, directs it to watch for the `file system full` message. The second column, which tells Swatch how to respond, might say `echo,bell,mail=sysadmin`. `swatch` echoes the message to the standard output, sends an audible signal to the terminals, and sends a notification email to the user sysadmin. More detail on `swatch` is included in Chapter 9.

Patches and Security

The first thing you want to do to be sure that your system is secure make certain that you are at the latest patch levels for your operating system. A `showrev -p` command will list the patches that are currently installed. As emphasized in Chapter 5, regular but guarded (install only what you know you need) installation of patches will go a long way toward protecting your systems from discovered security weaknesses.

In addition, information in the output of the `uname -a` command will tell you the "patch level" of the system. In the output below, the bolded characters display the patch level. It corresponds to the latest set of recommend patches installed on this system. Refer to Chapter 5 for additional information on patching Solaris systems.

```
# uname -a
SunOS boson 5.7 Generic_106542-18 i86pc i386 i86pc
```

Trusted Solaris 8

Trusted Solaris is a special release of Solaris that is able to properly process multiple sensitivity levels or "classifications" (e.g., secret and top secret) with respect to users' clearances and need-to-know. Much more like the standard Solaris product than trusted versions of Solaris in the past, Trusted Solaris 8 provides an environment for users of this labeled (often referred to as *classified*) information.

NOTE Currently, there is no Trusted Solaris 9.

One of the authors worked in the past for a highly security conscious federal agency that did not use Trusted Solaris. This was, to a large degree, because of the extreme lag in its availability following the release of each normal release. Trusted Solaris 8 came out close on the heels of Solaris 8 and represents a product much more in line with what system administrators are used to. This was far from the case in the past, when the cost and difficulty of using Trusted Solaris prohibited its use.

The primary benefit of Trusted Solaris 8 is that it makes the job of nailing down security on the system far easier than the normal operating environment.

Some of the features might not make much sense to readers who haven't worked with classified information or in environments where security labeling is a part of everyday life. Most corporations don't get beyond "proprietary" as a data label, and even this is not handled with significant diligence. The divide between unclassified and top secret information, on the other hand, is huge and serious.

Trusted Solaris 8 meets the security requirements of B1+ (a federal security classification that identifies the requirements for handling security-labeled information). It provides labeled security and better control of the policy of least privilege. It also provides rule-based access control (RBAC), described earlier in this chapter and also included in Solaris 9.

No longer is root a monolithic and all-powerful user. Trusted Solaris 8 divides superuser power into multiple roles and provides more powerful auditing so that you can easily track what users are doing. There are four default roles: security administrator, system administrator, primary administrator (who creates rights profiles and has the most abilities on the system), and oper (the operator). These are described in Table 13.11. System administrators must sign in as themselves and then assume the appropriate roles through RBAC.

Table 13.11 Administrator Roles

ROLE	DESCRIPTION
root	Only used for initial installation and configuration. Afterwards, no one should be assigned this role.
admin	Used for standard administration tasks such as adding accounts and performing backups.
secadmin	Used for security tasks and decisions such as administration of labels and security-relevant attributes of users, networks, and so on. Can also modify default roles, but these cannot exceed those of secadmin.
primaryadmin	Used only when the secadmin role is not adequate for a task.
oper	Used for operator tasks, such as backups and management of printers.

Data labels are assigned by the administrator. Users can only access files that fall within the labels they are allowed to access. A user who has been cleared for *secret* information, for example, is unable to access files that have been labeled *top secret*. Labels also take on the attribute of *compartment*, a classification that deals with the particular subject matter rather than the risk associated with its compromise.

The Trusted Solaris environment acts as a mediator in all security-related events. It compares the labels associated with data or processes with the privileges of the user trying to access them.

Trusted Solaris is also designed to fend off certain types of attacks. Trojan horse attacks, for example, whereby an imposter binary dropped into a system directory by an attacker attempts to divert information to some other place, are nearly impossible to craft. This is so because the Trusted Solaris environment displays a special symbol called the *trusted path symbol*, a noticeable, tamper-proof emblem at the bottom of the screen. This symbol lets users know that the tool they are using is authentic.

Trusted Solaris uses two forms of access control. These are referred to as *discretionary access control*—control that is at the discretion of the owner and is composed of the traditional Unix access controls (file access permissions) and *mandatory access control*—the control provided by the system that pays attention to the labels associated with processes and data. This activity is more pervasive than you might first imagine. Even the `ps` command is affected in that it shows only those processes owned by the user.

Trusted Solaris 8 documents are available from http://docs.sun.com.

Summary

The best advice we have to offer on the subject of system and Internet security is that you take it seriously. Systems face real threats—from targeted attacks to mischievous vandals. For large and complex networks, a dedicated systems security person is an extremely good idea but, even if you can't dedicate someone to the security role, there are a number of steps you can take to reduce the risk, including:

- Turn off services you don't need—in the `/etc/inetd.conf` file or `/etc/rc?.d` directories.
- Replace vulnerable services with more secure tools.
- Configure the tools you keep for maximum security.
- Wrap your services to limit the users and hosts that can access them.
- Put a firewall between your systems and the outside world.
- Use IPsec to protect services.
- Track security updates issued by Sun and other security organizations (e.g., CERT).
- Patch your systems appropriately and periodically.
- Maintain log files.
- Monitor your systems for security holes and evidence of tampering.

CHAPTER

14

Implementing High Availability: Eliminating Single Points of Failure

You've bought your shiny new Enterprise servers with their extremely high levels of reliability, availability, and scalability (RAS). You've been given impressive mean time between failure (MTBF) numbers and an assurance from your SunService sales rep that your systems will never fail. Now, you have Internet applications that you're expected to keep up 24 hours a day, 7 days a week. What will happen if there is an outage? How will you keep your applications running and your data available while maintenance is performed? Have you considered that planned outages, just like any catastrophe that takes down your servers, are downtime? Maybe it's time to start looking for a high-availability (HA) solution.

Many people confuse high availability with fault tolerance. High availability delivers up to 99.5 percent uptime, while fault tolerance will gain you that extra half percent. Though 100 percent availability is an attractive concept, that extra half of a percent can push your costs off the scale. Unless you can't survive without round-the-clock uptime, a high-availability solution might be quite sufficient. With the proper architecture and planning, HA can bring peace of mind to you and your organization—without the considerably higher cost of a fault-tolerant operation.

There is also confusion over the difference between high availability and disaster recovery (DR). Disaster recovery is the ability to recreate the environment in the event of a catastrophic event, such as flood or hurricane. Most DR planning revolves around making sure that reserve hardware and system backups of primary systems are available at an alternate site in case the primary site becomes unavailable.

There are also many ways to implement HA solutions and many solutions that promise to redefine what high availability is going to mean in your environment. The more elaborate, the higher the price tag—maybe. Don't be discouraged by the range of choices; if implemented properly, any solution to increase availability will save you time and headaches.

In this chapter, we explore high availability from the ground up. We take a hard look at single points of failure and what you can do to eliminate them, but we also look at some high-powered cluster solutions that can provide a level of service unavailable until now.

The Mission Plan for High Availability

Planning for high availability is not rocket science. Simply take a look at your systems and applications and eliminate single points of failure. Consider each component in turn. Start with the power source, move to the server hardware and system software, proceed to connectivity, and then finish with each application.

It still amazes us that people will implement elaborate HA schemes and yet not use appropriate power protection. This puts the onus for providing a source of clean, adequate power on someone else—and adds risks to your hard work from elements beyond your control. Power is one of the most overlooked components of mission-critical systems. Don't make this mistake yourself. A good uninterruptible power supply (UPS) can make the difference between a normal business day and an extended outage. Most system administrators don't have the luxury of a hardened data center. Even today, many servers are located in office environments. If you are one of the system administrators who manages servers in the office environment and have to protect your systems, there are several rules you should follow. They may seem intuitive, but we have seen them ignored enough times to feel they're worth saying.

- Do not put all of your HA servers on the same UPS. A UPS failure should not take down your entire HA solution. If your setup includes two servers, two A1000 arrays, two monitors, and so on, put each set of components on its own UPS.
- Do not consider a system safe unless every power cord is plugged into a UPS. During a power outage, you want every system component to work—the monitors and tape drives are not exceptions.
- Test the batteries on your UPS devices to make sure you will have enough time to shut systems down gracefully. An underpowered UPS is almost as bad as no UPS at all.

To avoid single points of failure within your server hardware, use as much redundancy as possible. If you're using the Ultra Enterprise line of servers, you need enough internal power supplies to absorb the loss of one. Mirror your OS and swap disks. If multiple CPU boards are installed and you have only two CPUs, be sure to place one on each board. Mirror your disks across multiple controllers; that way, if you lose a controller you won't lose the data on both disks. When using SPARC storage arrays for your storage, mirror across arrays or, if you have only one array, throw in a spare optical module in each system.

If using an E10000 (or E10K), don't assume that failover within the box will always be an option. Cluster it with another server able to handle minimally the most mission critical domains, and preferably all domains, although with degraded performance. Within the E10K there are still single points of failure, such as the centerplane, which can render the entire box inoperable. Planning ahead is always a good idea.

WARNING **Don't try to use your system service processor (SSP) as one of your failover devices. It will not be adequate to do the job.**

For any specific piece of external hardware that is not redundant, such as a tape jukebox or modem, you should have interfaces ready and waiting that can accommodate those devices on your other servers. This might involve having a kernel ready, but a reboot with a -r (i.e., `boot -r`) with the device attached will serve the purpose. If you are moving tape drives or jukeboxes, running both the `drvconfig` and `tapes` command will make those devices immediately accessible.

Install additional network interfaces to be employed in case of adapter failure. A failover to the second adaptor can be accomplished easily and will allow you to postpone repairs indefinitely; a couple of `ifconfig` commands—one to shut down the malfunctioning interface and one to start up the second—and moving the network cable and you're back in business. Please don't consider multiple ports on a QFE (Quad Fast Ethernet) board sufficient. Chances are, when you lose one port, you lose them all. Also, for completeness, primary and alternate interfaces should be plugged into separate network switches or hubs. If the hub or switch is the problem, you could have your system looping between adapters with no way to get messages out.

For example, if there are two hme (hundred-meg Ethernet) interfaces on a system, one live hme0 with IP address 1.2.3.4 subnetted to a class C, and one dormant hme1 that is available but not configured, the commands needed to migrate the IP address from hme0 to hme1 would be:

```
/usr/sbin/ifconfig hme0 0.0.0.0 down
/usr/sbin/ifconfig hme0 unplumb
/usr/sbin/ifconfig hme1 plumb
/usr/sbin/ifconfig hme1 1.2.3.4 netmask 255.255.255.0 up
```

If hme1 were already live with another address on the same subnet, hme1 would be replaced with hme1:1, since Solaris allows up to 255 virtual addresses per adapter.

One can argue that if failover software is used, these redundancies are not needed. While HA software will provide proper failover within a specified time frame, the best HA solution is one where system failover never happens. In other words, make each system in itself highly available, then couple these independently highly available systems with other highly available systems. Regardless of how good HA software is, in most cases that software will be restarting on an alternate system, and all network connections will be severed and system states will not be retained. By having redundant components within the system itself, the system can heal itself without your having to take down any of the running applications.

You must determine if you will need any of your own scripts, programs, or third-party utilities during a failover. If so, these should either be located on shared disks, copied to all HA servers, or mirrored to the other servers. For products requiring license keys, ask your vendors for a key to be used only in case of a failure. For a tape jukebox and backup software, be sure you can use any of the servers in the HA configuration as the backup server.

In regard to your applications, decide which ones are mission critical and should be part of the HA environment. Determine their system requirements. If the worst possible calamity happened and everything was failed to a single server, could that server handle the load? Nothing should be hard-coded in any application that would preclude its operation on another server.

Once single points of failure have been eliminated, it will be time to get the hardware and software fully prepped for a smooth failover (i.e., a takeover of data and applications elsewhere on your network) when a server fails completely or partially.

HA Configuration

Originally, HA solutions were all one-into-one configurations, each server requiring a twin server as an idle hot standby. Only complete system failovers—all services migrating to the standby—were performed when any failure occurred on the primary server. A one-into-one configuration is depicted in Figure 14.1. Today, there are many types of HA solutions to choose from that can provide M-into-N, symmetrical, and asymmetrical failovers. Figure 14.2 depicts a many-into-one configuration.

An asymmetrical HA configuration consists of two servers, with failover occurring in one direction only. Symmetrical HA involves possible failovers in either direction. M-into-N involves M services running on N servers, with any of the services able to fail over to any of the other servers. HA packages all perform some sort of clustering. *Clustering* can be defined as the combination of multiple systems with shared resources (see the *Cluster Servers* and *Introduction to the Cluster Grid* sections later in this chapter for more information on clustering). In most cases, those shared resources are disk space and IP addresses. Most clustering HA packages have two major facets to their operation: heartbeats and service groups. The *heartbeat* provides the connection between all of the systems in your HA cluster. Most heartbeats require at least one private network—although having two such networks is recommended, both for redundancy and for the elimination of the heartbeat as a single point of failure. Some HA packages utilize multicasts or broadcasts to determine the members of the cluster. We recommend, therefore, that you not place heartbeats on your public network, lest your corporate network evangelist club you over the head with a sniffer. Also, if you cannot use a separate hub for each private network, crossover cables are recommended for one-into-one configurations, because a single hub would constitute an unacceptable single point of failure.

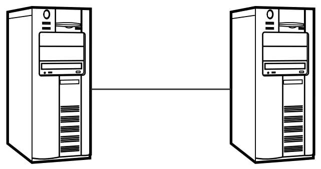


Figure 14.1 One-into-one.

For partial failovers (i.e., only some services fail), applications are split into service groups. Each service group is a logical unit that can be failed over on its own. For example, a server that is running two instances of Sybase SQL server and one instance of Oracle and providing NFS home directory services would have four logical service groups. Each of these groups would be assigned its own IP address. Port numbers for the database instances should be kept distinct among all instances within the cluster.

All drives and volumes for each service group should be placed in their own disk groups if using Veritas Volume Manager. If you're using Solstice Disk Suite, make sure your metadevices for each service group don't share drives, and that you have enough configuration copies of the metadb defined. If this is a small HA configuration with dual-ported SCSI disks between two systems, the SCSI targets (logical unit numbers, or LUNs) must not conflict for the controllers that are dual ported. Also, when planning the disk layouts, keep in mind what will happen if one of those devices fails. If using multiple disk arrays, try to keep each side of a mirror on one array. The authors have seen cases where volumes consisted of disks across two arrays, then the mirror of that volume was across the same two arrays. When one array failed, both sides of the mirror were toast. In a shared disk HA configuration, you still need something to share in order for failover to work. Keep this in mind when applications start requiring additional disk space, or business areas won't want to pay for the added costs of mirrored disk. Disks are cheap, lost business is not.

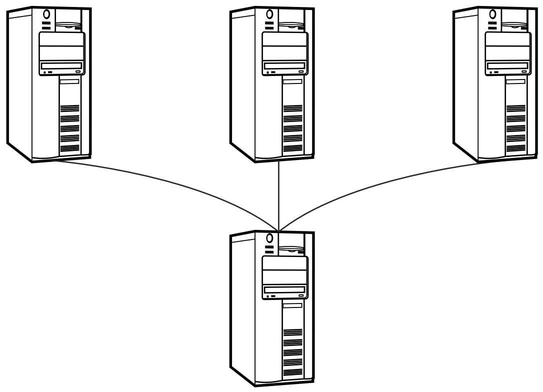


Figure 14.2 Many-into-one.

The key to service groups is remembering that IP addresses need to be distinct for each service. No multiple adapters? Don't worry—Solaris supports up to 255 virtual IP addresses for each adapter. For those who don't know, the syntax for starting up a new virtual address is `ifconfig interface:virtual# IP-address`. For example, `ifconfig hme0:1 191.29.71.48` starts up a second virtual address for the hme0 interface.

Why would we want to fail over a single service? There are a number of good reasons. Let's examine two examples:

In the first example, a controller failure on an application's primary system (machine one) causes a failover of that one service to the secondary system (machine two). Performance will be much better if applications not affected by the failure continue running on machine one than if all services fail over to machine two. By allowing a single application to fail over, hardware maintenance can be scheduled for machine one during a noncritical period when the rest of the services can be failed over with little consequence.

In the second example, you are testing a new operating system. If you have an asymmetrical configuration with two machines—one production and one test—with dual-ported storage between them, the operating system can be upgraded on the test box and each service failed over individually to methodically test how each application runs on the new operating system.

Rolling Your Own—Can It Be Done?

Once you've built redundancy into your servers, determined which applications are critical, and thoroughly thought through your failover scenarios, your environment is ready for HA. With the purchase of Open Vision by VERITAS, there are now three major vendors of HA software for the Sun Solaris environment: Legato, Sun, and VERITAS. Each of the HA packages offered by these companies has strengths and weaknesses. Should you consider rolling your own? Is it possible? Is it worth doing?

The first thing you need to determine is what type of availability is really needed. If you have mission-critical applications, buy the software. If, on the other hand, you can get by with modest lapses in service, but want to improve the uptime of non-mission-critical projects, then it is possible to create your own HA on a shoestring.

But while it is possible, we don't recommend it. Buy the software if you have the budget. If you cannot buy the software, consider whether you have the time to write and support a custom solution. This is probably the time to bring management in to help weigh the pros and cons of in-house development of an HA solution. One major plus for purchasing an HA package is the ability to point a finger and know whom to call when there is a problem.

WARNING **If you are planning to develop your own HA solution, a lab environment is highly recommended to test and verify your work. A misconfigured or misbehaving HA product can wreak havoc on an otherwise well-functioning network. You should insist that you be given opportunities to test your failover routines in an environment where no one else will be affected.**

If you decide to proceed with an in-house solution and feel that you know the risks, here are the major components you will need to implement in your HA solution:

- The ability to start, stop, and test each service from all servers.
- The ability to bring physical and virtual network interfaces up and down and to add and delete virtual IP addresses.
- The ability to import, export, and start disk groups and volumes.
- The ability to monitor the status of service groups and servers.
- The ability to send notification to system administrators or operators.

How much downtime and manual intervention you are willing to accept will determine how much functionality you need to implement into your HA solution. If you can accept a manual failover, you can simply write a script to import the drives and disk groups, configure the interface, and start the application. Procedures can then be written for operators to run these scripts as required. Figures 14.3 and 14.4 show a user oblivious to a failover that moves his access from one server to another. Do you need this level of seamlessness or can failovers be allowed to cause brief interruptions in service?

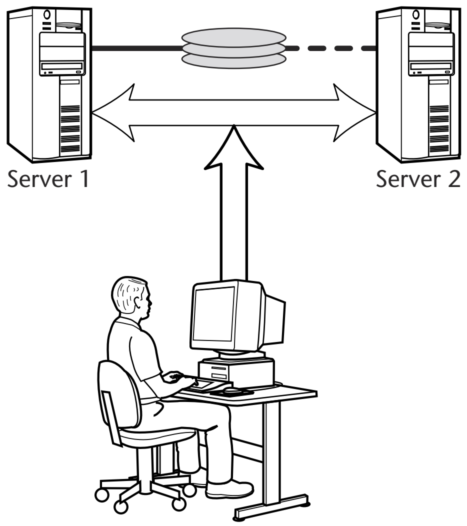


Figure 14.3 Before failure.

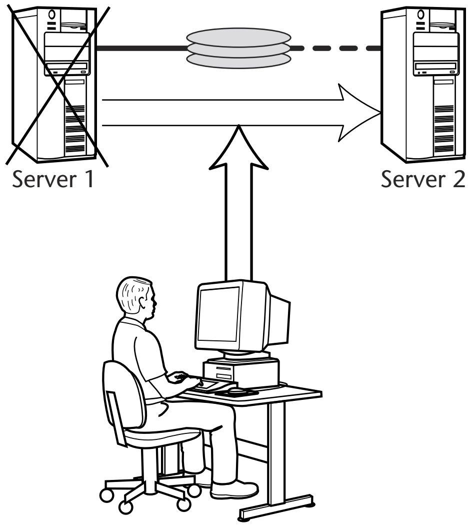


Figure 14.4 After failure.

Any of the components listed can be built via scripts. Most applications you are running should already have start scripts (see Chapter 4). Developing methods to determine whether an application is up and how to bring it down (hopefully other than `kill -9`) should be rudimentary. Using `ifconfig` to start an interface is also simple, but you don't want to bring up an IP address on one machine while it is still being used on another. Before you use `ifconfig`, `ping` the address and make sure that you don't get a response; if you get a response, the address is in use. The same goes for the importing of disk groups: VERITAS Volume Manager makes it quite easy to import a disk group (`vxdg import disk group`) and start volumes (`vxrecover -g disk group`). It even has a force option (-f) to ensure that the group is imported. To make your HA solution, the group needs to be exported when the application is brought down on the original server, if possible, and imported file systems should be `fsck`ed prior to being mounted.

Probably the toughest task in rolling your own HA system is the communication between the servers. Determining what is up, what is down, and who holds the ball for each service is not as simple as it sounds. For reliability, this should be accomplished across the private network. For each service, you need to maintain where the service is currently being run and the machine to which it is to fail over in case of a failure.

Once your solution is in place, the hard part begins. HA is now a part of your system. Any system upgrade, patch, or added application will most likely require modifications to both the other systems in the cluster and your failover software. Mirroring changes on all systems means that operating system levels and patch clusters need to be synchronized in order to prevent unforeseen circumstances.

TIP Test failovers regularly, both on a system and service group basis. The worst time to find out your HA solution doesn't work is when you need it.

The HA cluster can also be used to test applications in a new operating environment. This is recommended for use in development environments only, because changes can cause unforeseen consequences. In this scenario, a node in the development cluster would be backed up and then the OS would be upgraded or patched to the new releases for testing. Then, applications can be failed over to the new node and tested on the new OS or patch level and failed back. This works well in most cases. Beware of incompatibilities in RAID software such as VERITAS Volume Manager or Online Disk Suite. Keep good backups in case the testing corrupts the data.

Choosing a Third-Party HA Solution

When choosing a third-party HA vendor, you should consider all the options prior to making a decision. As we mentioned earlier, each vendor's product has pluses and minuses. Here is a list of questions that you should ask of each HA vendor:

- Is the installation simple? Can a cluster be installed in hours, days, or weeks?
- Are the test intervals customizable?
- Does the solution provide for symmetric, asymmetric, or N-node cluster capabilities?
- Are service groups supported, or is it system-level failover only?
- Does the solution support the servers and OS versions you are currently using or plan to use in the future?
- Are kernel modifications required to implement the solution?
- What types of disks and RAID software are supported?
- Can the solution be customized for in-house applications?
- What type of modules or agents are available (Sybase, Oracle, NFS, or HTTP)?
- Is root access required for all access to the applications?
- How is planned application downtime handled?

After these questions have been answered, you can make an intelligent informed decision on which vendor to choose. The authors have direct and indirect experience with both Veritas and Sun cluster products and find them extremely well engineered and fully functional. However, we would not recommend installing and using either of these products without proper training and an adequate startup period because they represent a different way of thinking about and configuring systems.

Implementing Your Solution

You have chosen a vendor and are ready to proceed to implement the solution. Remember the old adage—KISS (keep it simple, stupid)? Try to keep your configuration as clear and understandable as possible. The more complex the solution, the greater the chance for error. If you hire a consultant to implement your solution, be sure that internal employees also understand exactly how and why things are being set up. Establish procedures for anyone needing to interact with the HA system, so that mistakes can be confined to a minimum. Look for products that complement the HA package—like paging software. Finally, test all permutations of failovers on a regular basis. We can't reiterate this point enough. An untested HA solution is probably a nonworking HA solution. Besides, you and other staff members will likely feel more comfortable with your solution if you have experience with it.

Cluster Servers

One high-availability technology that has changed the character of system administration—not at the back end, but at the application layer—is clustering. Clustering breaks the one-to-one relationship between each service and the machine that it runs on, providing levels of availability that are astounding, while inserting a layer of complexity between the service offering and the systems that support it. While clusters make it possible, for example, for a service that is offered on any number of systems to perform as if it were a single extremely robust service running on a single system, it is the system administrator who needs to define how services are grouped and monitor cluster performance.

Though clustering technology is not new, with VAX clusters having made an appearance roughly two decades ago, it has gained quite a bit of momentum and made great strides into virtual computing in the last few years as increasing attention is being paid to high-availability architectures and the demand for nearly continuous uptime has become the rule more than the exception.

As the subtitle to one of Sun's documents expresses it, "Manage the Service, Not the Server." With the installation of clusters and cluster server software, system administrators' focus moves from thinking of their responsibilities as related to managing a group of *servers* to focusing on the *services* they are managing.

In this section, we will introduce important concepts related to clustering and then provide a brief introduction to each of two cluster server products—Veritas Cluster and Sun Cluster.

Cluster Concepts

A *cluster* is a group of systems in a loosely coupled relationship that together provide a single service to a set of clients. The service offerings can include almost any network-based services, such as databases, Web servers, print services, and so on. The following list briefly describes the major cluster concepts:

Highly available. A term used to describe services or systems that include some level of redundancy that minimizes downtime. In other words, some component can fail, yet the service or system continues servicing requests. Almost any component can be redundant in a highly-available system. This includes disks, controller, cables, power supplies and so on.

Scalable. A term used to describe the ability of an application or service to be expanded across a larger number of nodes without incurring a loss of manageability.

Fault tolerant. A term that is generally used to describe hardware, and describes error-resistant systems. Fault tolerance can be achieved in a number of ways, but often involves implementing more complex circuitry rather than simple redundancy. Most clustering systems use both redundant hardware and special software to manage the redundancy.

Fault detection. The process by which failure in a cluster is noticed so that some action can be taken to relocate the service and keep the service up.

Failover. The process by which cluster software relocates a service, generally after some failure on the initial system or component. Failover is a key component of cluster technology.

Failback. A term used to describe the situation in which a service, which has been transferred to a second node following a failure, is transferred back to the original node when that node comes back online.

Multihost disk. A disk that can be controlled by any of a number of nodes to which it is connected. Often, multihost disks used in clusters are RAID devices.

Cluster time. A term used to describe the required time synchronization between nodes in a cluster configuration.

Global namespace. A term for the naming of devices in a cluster. For example, a local disk might be called `/dev/dsk/c0t0d0s0`. The same disk with respect to the cluster might be referred to as `/global/.devices/node@d10/dev/dsk/c0t0d0s0`.

Cluster file systems. File systems that are mounted on all cluster members.

Split brain. A partitioning of a cluster caused by a failure.

Amnesia. A condition in which a cluster starts with old (not current) data.

Volume managers. Volume managers (see Chapter 11) are software products that provide a layer of flexibility and control between file system commands and the underlying disk structure.

Data service. An application that runs on a cluster.

Fault monitor. A service that probes data services to ensure that they are operative.

Resource group. A collection of resources grouped so that they can be managed as a single service.

Load balancing. An arrangement whereby any member of a group of servers can respond to a client's request for service. A *pure data service* allows any client request to be passed on to any server. A *sticky service* maintains a one-to-one relationship between a client and server for the duration of a session. Certain applications (e.g., Web applications that require a user login) depend on stickiness to function properly.

Cluster Components

A cluster is generally composed of such components as:

- Independent systems with local disks
- Shared disks or shared network storage
- Cluster interconnects (a private connection for communication between systems involved in the cluster)
- An administrative console for centralizing management of the cluster
- Public network interfaces for client communications

Key to cluster servers is the failover operation—the ability of the server to relocate a service in case of failure—and shared storage, allowing for data to remain current even when the controlling system has just been replaced.

Veritas Cluster

Veritas Cluster Server (VCS) provides basic application level failover. To explain this as simply as possible, we might say that this means that an application that has failed on one server can be migrated or *failed over* to another automatically. The method by which this failover happens, on the other hand, is anything but simple. The process of monitoring the health of applications, managing dependencies within a service (e.g., a database depends on a working network interface to be functional), identifying a candidate system to support a failed application, and managing the failover process is quite sophisticated.

This section introduces VCS technology. It explains how members of a cluster communicate and the service that this communication provides. It describes service groups and shows how service groups fail over. It describes resource dependencies and how they affect the way services are started and shut down. We also introduce the concept of system load and capacity and explain how these measures are used to determine which server is best suited to host a failed service. Last, we provide information on the directory structure used to house the VCS software.

Cluster Communication

Servers that are part of the Veritas cluster communicate with each other using special protocols that require minimal overhead. The VCS uses cluster communication for two purposes:

- To determine the state of each system defined as member of the cluster
- To gather data that describes the state of all cluster resources

On each system, there are agents that monitor the status of resources. The agents communicate the status of resources to the VCS engine known as the *high availability daemon* (`had`). It is `had` that uses the Group Membership Services/Atomic Broadcast (GAB) mechanism to communicate the status of all resources to other members of the cluster.

The lower-level transport protocol that runs underneath GAB is called Low Latency Transport (LLT), and runs directly on top of the Data Link Provider Interface (DLPI).

The main function of GAB is to insure that all members of the cluster have real-time information about the status of all resources.

The main function of LLT is to provide fast kernel-to-kernel communications to deliver data across one or more network links. LLT is also responsible for redirecting traffic to other links in case of a failure of the link that LLT currently uses. Another important function of LLT is performing heartbeat checks.

The Role of the Heartbeat

In VCS, preserving data integrity is the number one concern. Although the goal of the high availability daemon is to provide highly available service, any time there is a risk of data corruption, VCS will *not* bring up any service until it knows that the data is safe. In order to insure high availability while maintaining data integrity, VCS makes use of heartbeats. Two types of heartbeat are available in VCS: network heartbeat and disk heartbeat. For proper operation, a minimum of two heartbeat links is required. When there is only one heartbeat remaining in the cluster, there is no means to determine whether a failure is due to network connectivity or system failure. If the application that was detected as having failed is writing to shared disks, we should *not* bring the same application up on another system, since the failure could be due to a network connectivity problem. Otherwise, we might end up with two applications that simultaneously write to the same disk areas, resulting in data corruption. As a result, when there is only one heartbeat remaining in the cluster, VCS put the systems in *jeopardy mode* where no failover of any application is allowed until another heartbeat link is established. In other words, jeopardy mode safeguards the data. Figure 14.5 illustrates the role of the heartbeat.

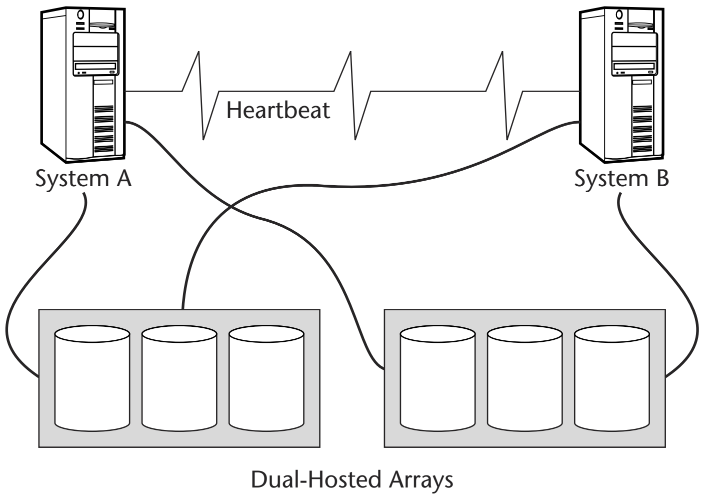


Figure 14.5 The network heartbeat.

Though it is extremely unlikely, it is possible that all heartbeat links could fail simultaneously. If this were to happen, each member in the cluster would believe that the service had failed on all other nodes and would attempt to start up the service. This is called the *split brain* condition, and data corruption is imminent. To make this condition even more unlikely, more network heartbeat links or disk heartbeats should be created.

VCS Service Groups

Veritas Cluster Server manages service groups composed of hardware resources such as disk drives and/or network interface cards, and software resources such as the Web server process (`httpd`) and/or file server process (`nfsd`). Administrators have the responsibility of defining service groups that are to be monitored and controlled by VCS so that the services within those service groups will have maximum availability.

As an example, consider a system that provides a database service. In order to provide such a service, the system needs a network interface card (NIC), an IP address, a physical disk, a file system on the disk to contain data files, and database server software running. If this database service needs to fail over from one system to another, *all* the resources needed to provide this service must migrate from the failed system to the system where the database service is being recreated. To accomplish this task, a system administrator would create a VCS service group that would include all the hardware resources (the NIC and the physical disk) and the software resources (IP address, file system and database server) necessary to provide the database service.

There are two categories of VCS service groups: parallel service groups and failover service groups.

Parallel service groups. These are used for applications that can run on more than one system in the cluster. For instance, if a service group is composed of a network interface card (NIC) and an IP address, this service group will run on all systems in the cluster since every node must have the NIC configured with the IP address in order communicate with other hosts on the network.

Failover service groups. These are used in situations where there is a constraint such that a service group can only run on one system in the cluster. An example of such a service group is the one that mounts a file system from disks in the shared drive and has a process that runs to perform read and write on that file system. If a failover service group is online on more than one system for any reason, there is a high risk that the data on the shared disks would be corrupted because more than one system would be writing to the shared disks at the same time.

Consider again the database service example mentioned previously. This service involves a listener (a process that is ready and waiting for service requests) and the database server itself. The database service interacts with the file system (see Figure 14.6) to access and update data while the file system accesses the physical disks. The database server also communicates with the IP protocol, while the protocol communicates with the network interface card. The database service, in other words, comprises all of these components.

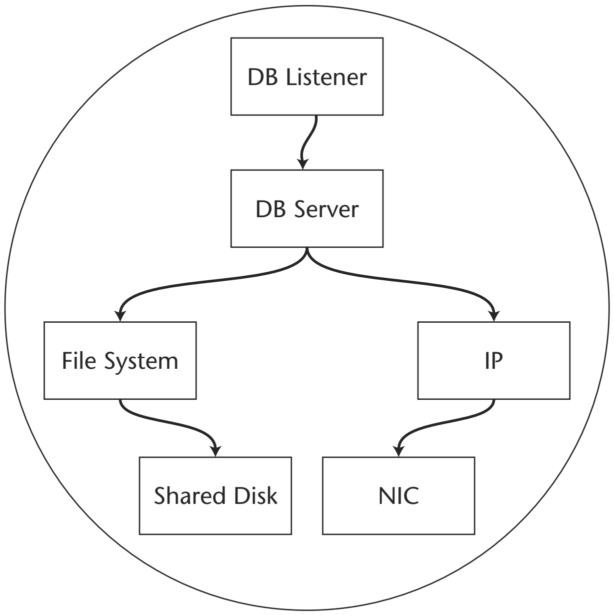


Figure 14.6 Database service group.

Let's assume that the database service is currently running on system A. To migrate this service group to system B, VCS will first bring down the DB listener process, then the DB server process. Once the processes are down, VCS will dismount the file system from the shared disk and bring down the IP address on the NIC. At this point, both the shared disk and the IP address are available for use by system B. VCS will then instruct system B to mount the file system that contains data files from the shared disk and assign the IP address to the NIC on system B. Once that is done, VCS will start up the DB server process and finally the DB listener process. External users will perceive a short downtime of the database server when the migration of database service from system A to system B occurs.

The general rule is that resources within a service group are brought online from the bottom up when the service group is brought online, and they are brought offline from the top down when the service group is brought offline.

Resource Dependency within a Service Group

Resources within VCS are organized in a hierarchical tree based on their relationships. For example, if the Web server software is installed on `/opt/apache`, then it relies on the file system `/opt/apache` to be up in order to run. To accommodate this relationship when working with VCS, we define the Web service resource as the *parent* of the mount point resource `/opt/apache`. In turn, the mount point resource `/opt/apache` depends on the disk `/dev/dsk/c0t2d0`, on which the file system `/opt/apache` was created, to be available. Therefore, the mount point resource `/opt/apache` is said to be the parent of the disk resource `/dev/dsk/c0t2d0`. In this convention, resources that are on the bottom of the hierarchy are the ones that other resources depend on, and resources that are on the top of the hierarchy are the ones that depend on some other resources. This dependency relationship is illustrated in Figure 14.7.

Resources can also be independent of each other, such as the Web server process and the network file system process, because the Web server software could be installed on a separate disk volume from the ones used for file sharing. There are also resources that are independent of each other from a hardware dependency standpoint but depend on each other from a functionality standpoint. For instance, the Web server process does not depend on the database server process to run, but if the Web service is used as a user interface to access data on the database server to perform transactions, then it would rely on the database server process to be running. System administrators are responsible for defining service groups in such a way that the resources within a service group are brought online in proper order to ensure that all resources in the group are brought online each time that service group needs to be brought online on a system. In the preceding example, if we define a service group called `websg` that contains the mount point resource `/opt/apache` and the Web server process, we have to make sure the Web server is the parent of the mount point resource `/opt/apache` so that each time VCS has to bring the service group online it will mount `/opt/apache` before starting the `httpd` daemon. Failure to do so may cause VCS to start `httpd` before it mounts or while it is mounting the file system `/opt/apache`, resulting in a failure to bring the service group `websg` online.

Resource Faults and Failovers in VCS

Each resource in VCS is designated as critical or noncritical. The failure of a critical resource on a system will cause the entire service group to fail over to other members in the cluster. In this case, resources in the group are taken offline in a *top down* manner until all resources in the group have been taken offline. Then, the resources in the service group will be brought online from *bottom up* on one of the other systems in the cluster. There are rules that govern the way failovers should occur. These rules constitute the *VCS failover policies*.

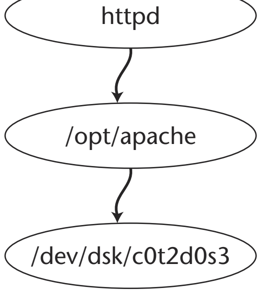


Figure 14.7 Parent-child relationship between VCS resources.

The failover policies are defined in the resource `FailOverPolicy` in the cluster's main configuration file, `main.cf`. There are three attributes to the `FailOverPolicy` resource: `Priority`, `RoundRobin`, and `Load`.

Each system in the cluster is assigned a priority value, with the lowest value representing the highest priority. If the policy is set to be `Priority`, the cluster server daemon will determine that the system with the highest priority will be the one to bring up the failed resource on another system.

If the policy is set to `RoundRobin`, the daemon will select the system with the least number of running service groups to run the failover application.

Finally, if the policy is set to `Load`, the daemon will go through a series of computations to determine the load on each system in order to designate the failover target. The system with the most available capacity will be the candidate. VCS computes a value for the capacity of the system by estimating the maximum load that the system can handle. Therefore, the term `AvailableCapacity` is defined as the capacity less the current load on the system. The daemon checks the value of this attribute and starts up the application on the system with highest value.

As an example of load-based failover, suppose we have three systems in the cluster: System A, System B, and System C. Let's assume further that at some point in time, the capacity, the load, and the available capacity on each system are as follows:

System A	Capacity = 200	Load = 120
System B	Capacity = 100	Load = 40
System C	Capacity = 250	Load = 210

Now let's assume that the Web service fails on System A. The cluster server daemon computes the `AvailableCapacity` of each system and obtains the following results:

SystemA	AvailableCapacity	200 - 120 = 80
SystemB	AvailableCapacity	100 - 40 = 60
SystemC	AvailableCapacity	250 - 210 = 40

Because System A is the system running the Web service when the service fails, the daemon will not consider System A to be a failover target. Between System B and System C, System B has higher available capacity. Therefore, the Web service group is started on System B. This scenario is depicted in Figure 14.8.

VCS Directory Structure on the Solaris Platform

VCS files are organized in the following three directory trees:

- `/opt/VRTSvcs`
- `/etc/VRTSvcs`
- `/var/VRTSvcs`

The directory trees are broken down in a fashion like that shown in Figure 14.9.

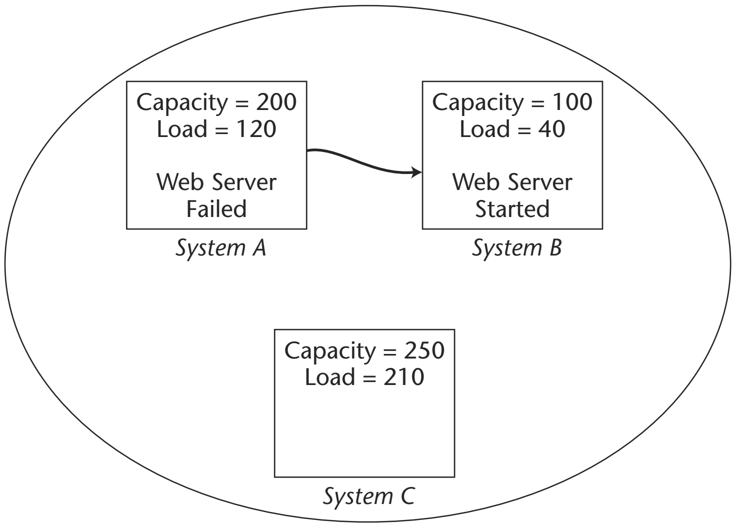


Figure 14.8 Load-based failover.

The directory `/opt/VRTSvcs/bin` stores executable binaries and shell scripts that start and stop the main process as well as secondary processes and agents. This directory should be added to the search path in the profile of the root user.

The directory `/etc/VRTSvcs/conf` contains resource type definition files such as `Types.cf`. In the subdirectory `/etc/VRTSvcs/conf/config`, there is a file named `main.cf`, which is the configuration file that stores information about the configuration of the entire cluster. This file is loaded into memory on each system in the cluster and is updated whenever a change is made to resource configuration. The update is done via the group atomic broadcast (GAB), which we described in an earlier section.

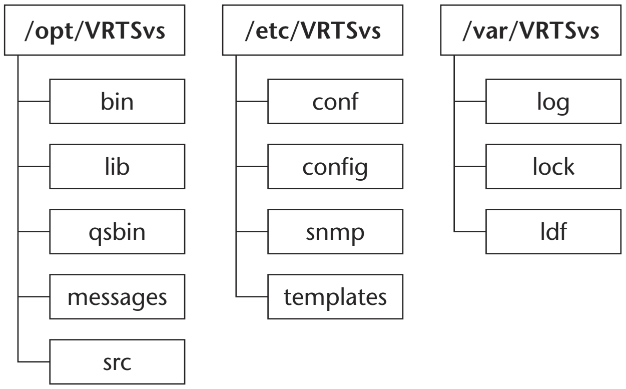


Figure 14.9 VCS directory structure.

The directory `/etc/VRTSvcs/snmp` contains the Management Information Base (MIB) for VCS. This MIB allows members of the cluster to send SNMP traps to network node managers so that all fault or failover events within the cluster can be sent to the central system-monitoring console. In order for the node manager to be able to interpret the alerts sent by the VCS notifier within the cluster, the MIB file in the directory `/etc/VRTSvcs/snmp` must be sent to the network administrator so that he or she can load it into the network node manager.

The directory `/var/VRTSvcs/log` contains various log files, including the main cluster log file, called `engineA.log`. Every event that occurs in the cluster is recorded into this engine log file.

Sun Cluster 3.0

Sun Cluster 3.0 is Sun's current offering in cluster software. Designed as an extension of Solaris 8, the product provides load balancing and consolidation of applications with the goal of keeping downtime to a minimum. It includes global devices, a global network service, and global networking.

Unlike prior versions of Sun Cluster software, many of the components in Sun Cluster 3.0 have become kernel agents or modules. Along with this change come a number of improvements:

- Up to six interconnects are supported (with no inherent upper bound).
- All interconnects are used simultaneously through dynamic load balancing.
- Interconnects can be enabled or disabled dynamically.
- Administration can be done through a GUI or command-line interface (CLI).
- Kernel agents provide zero copy optimization.
- Additional information on Sun Cluster 3.0 can be found at the Sun Cluster 3.0 Collection at http://docs.sun.com/db/coll/572.7/.

Cluster Communication

In a manner similar to that used by Veritas Cluster Software, Sun Cluster servers communicate by means of a *private interconnect* or *cluster interconnect*. These interconnects are physical network connections that are reserved for the sole use of the Sun Cluster software.

The private interconnects are used, as in VCS, for the heartbeat signals that ensure the connectivity between the nodes. However, with Sun Cluster 3.0, they are also used for application-level messages and data transfer for the software global features.

A minimum of two interconnects is required for resiliency. As was explained in the VCS section, a single connect leaves room for ambiguity in the case of certain failures. A maximum of six are supported, providing not only greater resiliency, but also for greater throughput (due to the way that the connections are load-balanced). These connections are *automatically striped*. That is, in similar manner to a striped volume, data can be sent in parallel across the available interconnects.

Network cards supported for use as interconnects include both 100baseT and Gigabit Ethernet. If more than two nodes are involved in a cluster, switches can be used to manage these interconnects.

The Cluster Membership Module

Within the interconnect infrastructure, cluster nodes that can communicate with the others are said to be *members* of the cluster. This communication is critical for distributing services. A special module called the Cluster Membership Module (CMM), ensures that only one cluster invocation is occurring at any particular time.

The CMM does the following:

- Maintains a proper view of cluster membership
- Forces a faulty node to leave the cluster until it is repaired
- Prevents partitioning of the cluster that could lead to the kind of troubles (e.g., split brain) described earlier

In addition to the split-brain problem, there is another problem referred to as *amnesia*, in which a cluster restarts with data that predates the shutdown. The CMM also guards against this problem.

The CMM in Sun Cluster 3.0 is a kernel module and, as such, is less vulnerable to resource problems that might affect the system in general.

Cluster membership is also determined by a voting mechanism. Each node within a cluster has a vote and additional "quorum disk votes" can also be configured. The voting mechanism provides a way to protect shared storage. A majority of votes must be present for a cluster partition to survive.

The `rgmd` (Resource Group Manager daemon) manages the reconfiguration process when a new cluster membership is established.

Cluster Configuration Control

Configuration changes in a cluster must be managed so that data is not corrupted and outages are not sustained. Sun Cluster maintains a repository of configuration information to prevent these problems. Sun Cluster 3.0 uses the Cluster Configuration Repository (CCR), a series of flat ASCII files stored in the root file system. The validity of the CCR is decided by a voting mechanism that prevents amnesia from occurring.

A cluster boots as long as a simple majority of votes is present. Once booted, changes can continue being made as long as one node remains.

Global Devices

Regardless of which nodes devices are connected to in a Sun Cluster, every disk, tape, CD-ROM, Veritas, or Sun Volume Manager Volume becomes a global device and is assigned a unique name and major/minor device number pair. Once these values are assigned, these devices appear in the `/dev/global` namespace and can be used as if they were local devices.

Devices may have a single controller or may be dual-hosted. For those that are dual-hosted, only a single path to the device can be active at a time. Because of the global nature of the namespace, these devices can be accessed from any location in the same manner.

Global Network Service

Sun Cluster makes use of global IP addresses (also referred to as a Global Interface, or GIF) to support scalable services. Although the global IP address is installed on a particular node, it is also associated with the loopback interfaces (e.g., `lo0:1`) of other cluster notes. This allows scalable services to come up on these nodes while still binding to the global IP address.

Incoming packets for the global service are then accepted through the global IP address and examined for the packet distribution policy of an ordinary, weighted, or wildcard sticky. As was mentioned earlier in this chapter, this distribution policy determines how connections are assigned. A weighted policy causes the packets to be hashed (on source IP and port), dropped into a bucket, and then transported over the private interconnect to the appropriate loopback. Sticky policies allow concurrent sessions over multiple TCP connections to share session information.

Global File Service

Sun Cluster's global file service provides a cluster-wide file service with a uniform directory namespace. The global file system is mounted identically on every node in the cluster, making the exact location of the data irrelevant to most uses.

Data Services

A data service is an application that runs on a cluster and is provided with failover capability through the cluster.

Resource Groups

Sun Cluster 3.0 provides three types of resource constructs—resources, resource groups, and resource types.

A resource type is actually a collection of programs, scripts, and files that provide the basics for how applications are started, stopped, and monitored. A resource is an instance of a resource type. Resource groups provide a grouping of resources and can be started and stopped together.

In similar manner to VCS, Sun Cluster 3.0 provides a dependency model that allows system administrators to specify what resources must be online before a service starts. Resource groups can also have dependency groups.

Failover and Scalable Data Services

A failover data service, in similar manner to a VCS failover service group, is one that can only be run on one node at a time. Two resource group properties—`maximum_primaries` and `desired_primaries`—are both set to one, indicating that only one such service can act on a single data set at any time. A scalable data service is similar to a VCS parallel service group and can run on any number of nodes at the same time.

Introduction to the Cluster Grid

Regardless of how much we might miss the days when we managed a group of services running on what was then an impressive Unix box and provided home directories and helpful scripts to our users, the service net of today and tomorrow is as different from this picture as a farm is from a metropolis. The emphasis has moved from servers to services and from hardware to resources. While high availability once referred to simple redundancy, today it refers to a dynamic environment in which services can float from one system to another, and resources can be shared among large groups of disparate users and system architectures.

What Is Grid Computing?

Grid computing provides a method of sharing resources across a group of users—from a small work group to one that is globally distributed—in a manner that is transparent to the users. The object of a grid is to maximize use of these resources. In the networks that the authors have managed in the past, resources were almost always allocated in a way that did not maximize their use. Some servers were overused and required nearly constant attention, while others housed resources that were rarely tapped. Some disks filled, while others sat idle.

Grids provide maximum usage of system resources by *virtualizing* them, allowing resources to be accessible across a large domain of users regardless of where or how they are logged in, by allowing resources to be controlled through policy, and by providing tools that facilitate monitoring and managing the grid.

Grids are divided into three levels of size and complexity, as follows:

Global grids. Provide geographically dispersed resources controlled by policies and protocols.

Enterprise grids. Allow groups of projects or departments to share resources.

Cluster grids. Allow a single group to use a particular resource or service.

Cluster grids are the simplest and smallest form of grid computing and are geared toward managing other resources—such as midrange servers, software clusters, and shared storage—in a heterogeneous computing environment.

Cluster Grid Architecture

Cluster grids can be thought of as comprising three layers: access, managing, and computation.

The *access layer* provides login and authentication services. Access can be lenient or tightly controlled, as needed. The *middle layer*, or *management layer*, provides the bulk of the cluster grid functionality. This includes file sharing (everything from home directories to compilers), licensing (allowing licenses to float across the grid as demand dictates), backup management (storage management services), and installation services (providing versioning and patching). The *compute layer*, or *computation layer*, provides jobs scheduling, message passing, and monitoring agents.

On Sun Today

One of the most exciting aspects about cluster grid technology is that software is available today for use on Sun systems. While grid computing has primarily been restricted to high-performance laboratories, it is headed toward the computing environments of large and midsized companies. In fact, the authors predict that you will be hearing a lot about cluster grids within the next couple of years.

Grid clusters allow the computing power of all systems on the grid to be managed as a large central resource, allows scheduling of jobs to be managed in such as way that resources and needs are matched to provide the best overall usage, and is ultimately scalable—allowing thousands of processors to participate in the work of the overall cluster.

The cluster software available from Sun today includes the Sun Grid Engine (SGE) and the Sun HPC ClusterTools software.

Additional features, important to the grid, are available through these tools:

- Solaris JumpStart
- Solaris Flash
- Sun Management Center (SunMC)
- Sun Validation Test Suite (SunVTS)

The Sun Grid Engine software provides job management services for both Solaris and Linux environments. These services include load balancing, job dispatching, accounting statistics, and the ability to suspend and resume jobs.

The types of logical hosts in an SGE setup include:

The master. The system to which requests are submitted and from which scheduling is managed. The daemons include `Schedd` (the scheduler) and `Qmaster` (the process that manages the queue, accepting requests and passing them on to the scheduler).

The shadow master. Any number of systems that monitor the master to ensure that it is working properly. A shadow master can assume control if the master fails. The daemon that runs on shadow masters is `Shadowd` (the process that monitors the master).

Execution. Hosts in the cluster grid that can execute jobs. The daemon on execution hosts is called `Execd`.

Submit. Hosts that submit, monitor, and administer jobs. No daemons are run on these hosts.

Administration. Hosts used to make changes to the cluster grid configuration. No daemons are run on these hosts.

The Sun HPC ClusterTools software provides a software environment for parallel applications, including high-performance libraries.

Cluster grid components are available from Sun today and at no cost. We encourage you to take advantage of the information and tools available. Please refer to the grid technology documents available at wwws.sun.com/software/grid/ for additional information.

Summary

High-availability solutions offer near-continuous availability of data and applications using redundant hardware and prepared software to accomplish or facilitate a smooth failover of services. You now know what HA is and isn't, how to plan for an HA configuration, and how to choose and implement an HA solution. These simple guidelines can help you create a highly available environment:

- Eliminate all single points of failure.
- Test failover regularly.
- Watch disk configurations and keep mirrors separate.
- Use redundant network components.
- Remember changes to one system should be applied to all systems in the cluster.
- Test failover regularly—especially after making changes. (It's important so it's listed twice!)
- Keep the design as simple as possible.

Technologies for providing high availability have come a long way since the first edition of this book. The increasing sophistication of cluster software servers, such as Veritas Cluster Server and Sun Cluster, are shifting the focus of systems management from servers to services. Grid computing is promising to provide computing resources across huge and heterogeneous networks. High availability has come of age.

PART

III

Looking after Your Hardware

If you are a typical Systems Administrator, hardware is something that you try to deal with as little as possible. If you are in a large corporate environment, chances are that you have Sun or a third party maintaining your equipment. No, this section is not a SunService Field Guide nor is this section a detailed explanation of what every piece of Sun hardware is and does. Rather, the eclectic collection of chapters in this part details items that will assist you with the software and methodologies required to install and troubleshoot your hardware and keep it running.

Chapter 15, deals with system commands used to troubleshoot your hardware and assist you in determining when a service call should be placed. Chapter 16 brings up a sysadmin's favorite subject: modems and printers. Because the first Sun systems rolled off the assembly line in 1982, serial ports and the configuration of devices connected to them has always been a challenge. This chapter is meant to assist you in understanding how to install and troubleshoot these pesky devices.

The following two chapters introduce some of the most impressive hardware in Sun's line-up—the Starfire, covered in Chapter 17, and the Sun Fire server line, which is covered in Chapter 18. In these chapters, we describe in detail the hardware of these amazing servers as well as methodologies and commands used to install, implement, and troubleshoot them. Though these servers function like other Enterprise boxes from an OS standpoint, they also have many other features and peculiarities that make their administration unknown territory for even the most seasoned Solaris administrators. Extremely scalable, yet temperamental machines, these servers redefine the term "high-end." If you support Starfires or Sun Fire servers in your corporate environment, or just want to learn about these fascinating architectures, these chapters are a must read.

CHAPTER

15

Maintaining Your Sun Hardware

While many of us use Sun Service or other third-party vendors for our system maintenance, it is becoming increasingly important that we understand the details of what our hardware is and does. This knowledge is crucial if we are to assist our vendors in troubleshooting problems when they occur. A disturbing trend among service companies is to have a clear separation between service technicians who support software and those who support hardware. Most system administrators, working in the trenches, realize that problems are not always that cut and dried, and have quite a bit of experience bridging the divide between software and hardware.

This chapter discusses maintenance procedures that should be followed to keep your equipment running at peak efficiency and details what some of the more mysterious hardware is. Also, a long section on NVRAM should tell you everything you've ever wanted to know about the machine identity and hardware parameters for many Sun systems.

Device Names

To make full use of the information presented to you at the OBP, in the system messages (`/var/adm/messages`) files, and in the output of certain commands you will need to know how the devices relate to the devices on your system. An identifier such as `pci@0,0/pci1014,22@9/pcie11,4030@0/cmdk@0,0:a` might or might not refer to the device known as /dev/dsk/c1d0s0. How do you know and where do these names come from?

The Solaris kernel is autoconfigured at boot time. The system determines what devices are attached, using the POST and OBP, and loads the required modules into memory. Device drivers are loaded the first time each of the corresponding devices is accessed.

Logical device names—those names that we sysadmins use to refer to devices (such as /dev/dsk/c0t0d0s0), both interactively and in configuration files such as /etc/vfstab—are actually symbolic links to the physical device names. In fact, you can see this clearly if you do a long listing of such a device. You'll see something like this:

```
$ ls -l /dev/dsk/c1d0s0
lrwxrwxrwx   1 root     root         59 Dec 19  2001 /dev/dsk/c1d0s0
-> ../../devices/pci@0,0/pci1014,22@9/pcie11,4030@0/cmdk@0,0:a
```

The *physical device names* in the /devices file structure are configured at installation time or with the drvconfig command. These file system entries actually point to the device drivers in the kernel.

Devices are also referred to by a third type of name called *instance names*. These names refer to the device driver that is used to control the device and an instance number that refers to one of a set of such devices. A device name such as sd0 or ssd1 would be an instance of a device controlled by the sd or ssd driver.

Where logical and physical device names are associated with physical names through use of symbolic links, instances are associated with physical names through entries in the /etc/path_to_inst file. The following entry, for example, associates the physical device /devices/pci@0,0/pci1014,22@9/pcie11,4030@0/cmdk@0,0:a with the instance 7 and the device driver cmdk:

```
"/pci@0,0/pci1014,22@9/pcie11,4030@0/cmdk@0,0" 7 "cmdk"
```

These instance names are used in the output of commands such as iostat, as seen in this example (notice the listing of cmdk7):

```
$ iostat
 tty        cmdk7           fd0           sd0          nfs1    cpu
tin tout kps tps serv kps tps serv kps tps serv kps tps serv us sy wt id
1   29   52  4   4    0   0   0    0   0   0    0   0   0    0  1  0  99
```

Troubleshooting Your System Hardware

Sun systems are renowned for running practically forever. The authors, in fact, have encountered numerous Sun systems that, aside from occasional OS upgrades, have run for 5 to 10 years without downtime. It is imperative, however, that the people administering Sun systems know how to test and maintain them. Many of the newer systems come with impressive diagnostics that not only will report hardware faults, but also will tell you the temperature of the boards in the system. This section provides an example of using the `prtdiag` command for troubleshooting, and then discusses both the OpenBoot PROM and NVRAM (or SEEPROM for Sun Fire servers)—the core of many SUN systems.

Operating Environment Troubleshooting Tools

The tools described in this section can be used for troubleshooting while the system is running normally. All of these tools are available within the normal Solaris installation or on the supplemental CD-ROMs.

prtdiag

The `prtdiag` command displays configuration, diagnostic data, and environmental information about the particular system. The contents of the output differs from one type of system to the next. Following is sample output from the `prtdiag` command on an E450. Note that, even if the hardware is thousands of miles away, you can get quite a bit of information about it by running this command in a Telnet (or `ssh`) session.

```
./prtdiag -v
System Configuration:  Sun Microsystems  sun4u Sun Enterprise
450 (2 X UltraSPARC-II 400MHz)
System clock frequency: 100 MHz
Memory size: 512 Megabytes
```

As you can see from the preceding output, this system has two 400-MHz processors and 512 MB of RAM.

```
========================= CPUs =========================
                   Run    Ecache   CPU     CPU
Brd  CPU   Module   MHz     MB     Impl.   Mask
---  ---  -------  -----  ------  ------   ----
SYS    1        1    400     4.0   US-II    9.0
SYS    3        3    400     4.0   US-II    9.0
```

The preceding output shows that there are two system boards (labeled 1 and 3), each with a single CPU module installed.

```
========================= Memory ========================
       Interlv.  Socket   Size
Bank    Group     Name    (MB)  Status
----    -----    ------   ----  ------
  0      none      1901   128      OK
  0      none      1902   128      OK
  0      none      1903   128      OK
  0      none      1904   128      OK
```

On this E450, there are four 128-MB dual in-line memory modules (DIMMs) installed. All of them are reportedly running fine (see output above). The IO cards, shown in the output also are reported to have no failures.

```
========================= IO Cards =========================
     Bus   Freq
Brd  Type  MHz   Slot  Name                      Model
---  ----  ----  ----  ----------------------    ----------------------
SYS   PCI    33     5   pciclass,001000          Symbios,53C875
SYS   PCI    33     7   pciclass,068000
SYS   PCI    33    10   SUNW,m64B  ATY,GT-B
No failures found in System
```

The following output shows that `prtdiag` has found nothing wrong with the system. Note that if NVRAM has been replaced, all failure information is cleared.

```
====================== Environmental Status ======================
       #       System Temperatures (Celsius):
     ------------------------------
     AMBIENT   23
     CPU 1     40
     CPU 3     40
     =================================
```

The temperature on the system boards is one of the more important things to watch. The higher the temperature becomes, the more the chance there is of developing a problem. If the temperature exceeds 50 degrees Celsius (122 degrees Fahrenheit), you run the risk of the system shutting itself down to prevent further overheating. Note that if your system keeps shutting itself down because of temperature, you have a cooling issue that should be resolved. When the temperature gets too hot, the displayed message will show up as `WARNING` or `CRITICAL`.

```
Front Status Panel:
-------------------
Keyswitch position is in Diagnostic mode.
System LED Status:    POWER     GENERAL ERROR      ACTIVITY
                      [ ON]         [OFF]           [ ON]
                    DISK ERROR  THERMAL ERROR  POWER SUPPLY ERROR
                      [OFF]         [OFF]           [ERROR]
Disk LED Status:        OK = GREEN      ERROR = YELLOW
                DISK 10: [EMPTY]        DISK 11: [EMPTY]
                DISK  8: [EMPTY]        DISK  9: [EMPTY]
                DISK  6: [EMPTY]        DISK  7: [EMPTY]
                DISK  4: [EMPTY]        DISK  5: [EMPTY]
                DISK  2:    [OK]        DISK  3:    [OK]
                DISK  0:    [OK]        DISK  1:    [OK]
=================================
```

This display is equivalent to the status lights on the box and will show errors when they occur. In the case of the preceding output, it looks like one of the power supplies is bad and should be replaced. An error is displayed. The following output provides some additional details on both fans and power supplies.

```
Fans:
-----
Fan Bank  Speed   Status
--------  -----   ------
CPU         49      OK
PWR         31      OK
Power Supplies:
---------------
Supply    Rating   Temp   Status
------    ------   ----   ------
  0        550 W   37      OK
  1        550 W   35      OK
  2        550 W   34      OK
```

The next section of output shows all of the hardware revisions of the boards in your system. Prior to performing OS upgrades, you should check to make sure there are no upgrades in hardware required as well. Hardware compatibility lists can be found at http://sunsolve.sun.com.

```
========================= HW Revisions =========================

ASIC Revisions:
---------------
STP2223BGA: Rev 4
STP2223BGA: Rev 4
STP2223BGA: Rev 4
STP2003QFP: Rev 1
STP2205BGA: Rev 1
FEPS: SUNW,hme Rev c1
System PROM revisions:
----------------------
  OBP 3.12.2 1998/09/14 11:28   POST 6.0.6 1998/07/16 18:40
```

The OpenBoot PROM (OBP) level is extremely important to correct system operation. Make sure that OBP levels are checked at least quarterly, because many of what seem to be hardware problems can be related to boot PROM levels.

Using SunVTS

Sun Validation Test Suite is a versatile tool for hardware testing that is included on one of the supplemental Solaris CDs. The GUI interface that is used to select the tests and number of repetitions that should be run can be displayed and used on a remote system so that you can control the execution of fairly exhaustive testing from your desk of a server that was just brought back online after a failure.

SunVTS is a tool that was designed to support Sun systems and peripherals and provides tools that allow system administrators to isolate the cause of system problems fairly quickly. The tools can be run within CDE, OpenLook, or terminal devices. Tests can be remotely scheduled and monitored. SunVTS is ideal for stress testing, and ideally should be run on a system that is not yet in use.

Using Sun Explorer

The Sun Explorer is a data collector tool. It uses a number of scripts and executables to gather extensive system information in a form that is meant to be analyzed by Sun and stored in a configuration database. More information about Sun Explorer is available at the following URL: http://sunsolve.sun.com/pub-cgi/show.pl?target=explorer/explorer.

prtfru (Sun Fire)

The `prtfru` command obtains field-replaceable unit (FRU) information from a system and displays it in a treelike structure. It can be used to display the contents of the FRU SEEPROMs. Information provided by this tool includes:

- FRU description
- Part number and serial number
- Hardware revision levels
- Temperature, voltage, and power data

prtpicl (Sun Fire)

The `prtpicl` command displays the name and *Platform Information and Control Library* (PICL) class of all nodes in the PICL tree. To display the high temperature and low temperature critical thresholds for each component, use the `prtpicl -v` option.

The following is partial sample output from the `prtpicl` command.

```
% prtpicl
  / (picl, 4300000001)
     SYSTEM (picl, 4300000005)
         MOTHERBOARD (picl, 430000000a)
             CPU0_PFAN_TACH (fan-tachometer, 43000000e5)
             CPU1_PFAN_TACH (fan-tachometer, 43000000ef)
             CPU0_SFAN_TACH (fan-tachometer, 43000000f9)
             CPU1_SFAN_TACH (fan-tachometer, 4300000103)
             IO_BRIDGE_PFAN_TACH (fan-tachometer, 4300000135)
```

We highly recommend the Sun Fire diagnostics and troubleshooting information available at this URL: http://docs.sun.com/source/806-6597-11/dak_sm_n.htm#998952.

Miscellaneous

Don't overlook the value of system messages files that may contain important information about hardware errors. The `/var/adm/messages` files should be examined periodically. We suggest using tools that summarize messages and/or highlight those that may indicate that serious errors are occurring (e.g., warning messages).

Open Boot PROM (OBP) Troubleshooting Tools

While at the ok prompt, there are many commands that you can use to troubleshoot your system. In this section, we cover many of these commands and offer some insights into how and when these commands might be useful to you.

To begin with, you should know the various ways that the STOP key can be used when booting your system (see the Table 15.1).

Also, if the `diag-switch?` parameter is set to `true`, output from the power-on self test will be displayed on a serial terminal. The `disg-switch?` parameter can be set to true with the following command:

```
diag-switch? = true
```

banner

The `banner` command displays the power on banner. It generally includes such information as the speed of the CPU, the OBP revisions, how much memory is on the system, and the hosted and Ethernet address.

probe-sbus

This command probes devices attached to the sbus. This only works if the system is sbus-based.

Table 15.1 Uses of the STOP key

KEY SEQUENCE	PURPOSE
STOP	By itself, STOP bypasses the Power-on Self Test (POST).
STOP-A	Aborts the POST.
STOP-D	Forces a diagnostic power on. The `diag-switch?` NVRAM parameter will be set to `true`. This does not work on USB keyboards.
STOP-F	Forces both the input and output to go to `ttya`. L1-A is still accepted on the keyboard. This does not work on USB keyboards.
STOP-N	Resets NVRAM settings to the defaults. This does not work on USB keyboards. If you are working on a Sun Blade 1000 or 2000, power up, wait for the power LED to blink and the speaker to beep, then hit the power button twice in a row.

probe-scsi and probe-scsi-all

The `probe-scsi` and `probe-scsi-all` commands probe your primary SCSI bus or all of your SCSI busses and report back on the device information that is reported back. This helps you to determine whether SCSI devices are recognized by the basic hardware and whether, in fact, they are communicating information with the system. A device that is "invisible" when probed through the OBP is clearly malfunctioning. SCSI probes should also allow you to detect SCSI target conflicts, allowing you to resolve the conflicts before booting your system. The rule about SCSI targets is simple—every device on a single SCSI bus must have a unique ID. Most external disks have external selectors. However, the authors have seen disks with SCSI target selectors that were, in fact, not connected to the disk inside the case. The SCSI target can also be set using jumpers on the drive itself. The `probe-scsi` commands will confirm your selection or tell you that a setting is not what it appears to be. The command output below, for example, shows you that the attached disk is set to SCSI target 3.

```
ok probe-scsi
Target 3
  Unit 0 Disk SEAGATE ST1480 SUN04245828 Copyright (c) 1991
Seagate All rights reserved.
```

One problem with devices on SCSI chains is that of termination. The typical and proper layout of a chain of SCSI devices is to have the faster devices closer to the CPU and the last device terminated. However, some devices may be terminated internally.

If you are using older SCSI hardware, you may have any of several types of SCSI connectors—from the original DB50 (SCSI 1), through the HD Centronics 50 (SCSI-2), the micro ZDB50 (SCSI 2), and the Ultra-wide SCSI or Micro DB68. Cabling these devices together can be annoyingly complicated if you have to search for the proper type of adaptor cable for each device.

probe-fcal-all and probe-ide

The `probe-ide` command is similar to the `probe-scsi` commands but is intended for IDE drives. For each IDE device that is connected and active, the target address, unit number, device type, and manufacturer's name are reported, as shown in the example below. The `probe-fcal-all` command discovers FC-AL (Fiber Channel Arbitrated Loop) devices—for example, disk arrays that use high bandwidth.

```
ok probe-ide
  Device 0  ( Primary Master )
          ATA Model: ST34342A

  Device 1  ( Primary Slave )
          ATA Model: ST34342A

  Device 2  ( Secondary Master )
         Removable ATAPI Model: CRD-8160B
```

test device and test-all

The `test device` command is used to run the self-test associated with a particular device. For example, you might test any of the following devices:

screen. Tests the graphics hardware, requires `diag-switch?` to be set to `true` (see the information that follows).

floppy. Tests the floppy drive, requires a formatted diskette in the drive.

net. Runs an internal and external network interface test; an active network cable must be connected to the port.

ttya. Sends test patterns to serial port a; output requires that a dumb terminal or other device be connected.

ttyb. Sends test patterns to serial port b; output requires that a dumb terminal or other device be connected.

keyboard. Tests the keyboard, flashes keyboard LEDs, and displays Keyboard Present message.

The `test-all` command runs through a series of tests, effectively testing each device that includes a self-test in device-tree order. It provides a quick way to do a hardware check of basic system components.

watch-clock

This tests the clock function.

watch-net and watch-net-all

The `watch-net` and `watch-net-all` commands are used to test network connections. `watch-net-all` watches each interface, while `watch-net` monitors the primary network interface. Each packet that is observed by the interface results in a dot appearing on the screen, while packets arriving with errors (e.g., CRC errors) appear as the letter X. If no activity is seen when one of these commands is used, your network connection may be malfunctioning or your network interface may be bad. Check your cabling and local hub if applicable. The authors always keep spare known-to-be-good network cables on hand to confirm or rule out problems with the cable when testing an uncommunicative system.

```
ok watch-net
Hme register test --- succeeded.
Internal loopback test -- succeeded.
Transceiver check  -- Using Onboard Transceiver - Link Up.
passed
Using Onboard Transceiver - Link Up.
Looking for Ethernet Packets.
'.' is a Good Packet.  'X' is
a Bad Packet.
```

```
Type any key to stop.
...............................................
..........................
ok
```

words

This displays all OBP commands and methods.

obdiag

The `obdiag` tool allows you to interactively run a number of diagnostic tests on any of the following systems:

- Sun Enterprise 420R Server
- Sun Enterprise 220R Server
- Sun Ultra Enterprise 450 Server
- Sun Ultra Enterprise 250 Server
- Sun Ultra 80
- Sun Ultra 60
- Sun Ultra 30
- Sun Ultra 10
- Sun Ultra 5

It not only tests the main logic board, but also these interfaces:

- PCI
- SCSI
- Ethernet
- Serial
- Parallel
- Keyboard/mouse
- NVRAM
- Audio
- Video

.speed

This command displays the CPU and bus speeds.

.version

This command displays the version of the OpenBoot PROM, OBDIAG and POST.

.properties

This command displays various information about a node. This includes its address and name. The variety of information displayed varies from system to system.

.enet-addr

This command displays the Ethernet (MAC) address.

.asr

This command displays system device settings and their status. For example:

```
ok .asr
System status :     Enabled
CPU0 :              Enabled
CPU1 :              Enabled
SC-MP :             Enabled
Psycho@1f :         Enabled
Cheerio :           Enabled
SCSI :              Enabled
Mem Bank0 :         Enabled
...
```

asr-enable and asr-disable

These commands enable and disable system devices, respectively. You can use the .asr command to determine the names to be used with this command.

show-devs

This command displays devices known to the system.

show-disks

This command displays the physical device path for disk controllers.

show-displays

This command displays the path for any frame buffers on the system.

show-nets

This command displays device paths for network interfaces.

show-post-results

The output of this command varies somewhat from system to system, but it summarizes the results of the power-on self test (POST).

show-tapes

The `show-tapes` command displays the path to tape devices.

.post

The `.post` command also displays the results of the power-on self test, as follows:

```
ok .post
System status :        OK
CPU0 :                 OK
CPU1 :                 OK
CPU2 :                 OK
```

Troubleshooting Resources

We recommend the following resources for gaining further insights into your hardware:

- The SunSolve Online Hardware Handbooks available at http://sunsolve.sun.com/handbook_pub.
- Information about patch and firmware updates available through http://sunsolve.sun.com/pub-cgi/show.pl?target=patchpage.
- The PatchPro tool for assistance with patch management and installation—available at http://patchpro.sun.com.
- *PatchPro Expert* can be used to analyze your system and generate a list of required patches. This tool requires a Java-enabled browser. Check the site for its availability. No information about your system is transmitted.
- *PatchPro Application for Solaris 9* can both analyze your system and download required patches. No information about your system is transmitted. It only downloads digitally signed patches on an encrypted connection.
- *Patch Manager Base 1.0 for Solaris 2.6 through Solaris 8* analyzes your system, downloads required patches, and installs them. Like PatchPro for Solaris 9, it only downloads digitally signed patches on an encrypted connection.

Upgrading the OpenBoot PROM Level

Upgrading your boot or Flash PROMs is as simple and painless as a tooth extraction. And, while you won't get sued for malpractice if you mess up, you can render your system totally inoperable and require a system board replacement.

E3X00/4X00/5000/6X00

If you are especially lucky, you'll have Ultra Enterprise machines. These are the only systems that can be safely upgraded *while* they are up and running. A reboot is required to activate the new firmware. You should write down the values of all of your NVRAM settings in case the variables are reset to their defaults during reprogramming. Also, make sure that the front panel key switch is in the secure position and that Jumper P601 is off. If the jumper is on, the system board's Flash PROMs will be write-protected, and you will see an error to this effect. The series of steps required to upgrade a system is as follows:

1. Download the update from SunSolve.
2. Execute the `flash-update` binary with the highest revision level. It will extract and install the flashprom driver on the system. Then, it runs the executable to start the update process.
3. The program displays the current revision of the PROMs in your system and the versions that are available, and asks whether you wish to proceed with the update. Answer yes (i.e., y) and the reprogramming begins automatically.
4. The program will then display its progress at every step.

Following is sample output where a machine has boards 0, 2, and 3, and its Flash PROMS are being updated:

```
# ./flash-update-<latest-rev>
Generating flashprom driver...
Generating SUNW,Ultra-Enterprise flash-update program...
Current System Board PROM Revisions:
------------------------------------
Board  0:        cpu OBP   3.4.2 1998/01/20 11:17 POST  3.0.3
1998/02/19 13:54
Board  2:        cpu OBP   3.4.2 1998/01/20 11:17 POST  3.0.3
1998/02/19 13:54
Board  3:  dual-sbus FCODE 1.7.0 1998/03/20 07:07 iPOST 3.0.3
1998/02/16 13:55
Available 'Update' Revisions:
-----------------------------
cpu OBP   3.4.3 1996/04/04 20:23 POST  4.1.4 1998/04/04 20:23
dual-sbus FCODE 2.7.0 1998/03/04 20:23 iPOST 4.1.4 1998/04/04 20:23
upa-sbus FCODE 2.7.0 1998/03/04 20:23 iPOST 4.1.4 1998/04/04 20:23
Verifying Checksums: Okay
```

```
Do you wish to flash update your firmware? y/[n] : y  <-- Enter y here.
Are you sure? y/[n] : y <--Enter y here.
Updating Board 0: Type 'cpu'
1 Erasing ... Done.
1 Verifying Erase ... Done.
1 Programming ... Done.
1 Verifying Program ... Done.
Updating Board 2: Type 'cpu'
1 Erasing ... Done.
1 Verifying Erase ... Done.
1 Programming ... Done.
1 Verifying Program ... Done.
Updating Board 3: Type 'dual-sbus'
1 Erasing ... Done.
1 Verifying Erase ... Done.
1 Programming ... Done.
1 Verifying Program ... Done.
```

The number displayed on the leftmost column is the pass number. If any of the four steps listed here fails, all four steps have to be repeated and the pass number will be incremented.

Desktop Flash PROM Update

For desktop machines (Ultra 1, 2, 5, 20, 30, 250 or 450), the process of updating the PROM is more painful than for Ultra Enterprise machines. Essentially, a new boot image needs to be installed that boots a single purpose kernel used for updating the Flash PROMs. To do so, follow these steps:

1. Download the appropriate flash files from SunSolve.
2. Copy the boot image to the `flashboot` directory and give it a name you can remember. Also set permissions on the directory as follows:

   ```
   # cp flash*latest  /flashboot
   # chmod 755 /flashboot
   ```
3. Now bring the machine down to the ok prompt then physically power off the system as follows:

   ```
   #init 0
   ok
   ```
4. Next, set jumper J2003 on each machine to write enable (pins 2 and 3) in order to enable updating of the Flash PROMs.

WARNING These jumpers should be touched only while the power is *off* and you are wearing an antistatic strap. Failure to follow these precautions could result in damage to your hardware.

5. Once the jumper is set and the machine is put back together, power the system back on and hit Stop-A when you see the banner. You should then be back at the ok prompt.
6. Issue the command `boot disk /flashboot`, replacing `disk` with the appropriate alias if you are not using `disk` as your disk alias. If you are booting over the network (i.e., `boot net`), the flashboot file should have been placed in the corresponding root directory for this client on the server providing the files. In other words, if the system machine1 is booting off of machine2, the flashboot file should be `/export/root/machine2/flashboot`.

   A version of the following menu will appear:

```
Standalone Flash PROM Update Utility, Rev. 2.0
  Ultra(tm) 1
  Ultra(tm) 2
  Ultra(tm) 5/10
  Ultra(tm) 30
  Ultra(tm) 60
  Ultra(tm) Enterprise(tm) 250
  Ultra(tm) Enterprise(tm) 450
This utility allows you to interactively update the firmware
revisions in specific system Flash PROM components.
Type h for help, q to quit, Return or Enter to continue: <= Hit Enter
here.
Every precaution should be taken to prevent the loss of system
power during the Flash PROM programming process!
Type h for help, q to quit, Return or Enter to continue: <= Hit Enter
here.
      Firmware Release(s)              Firmware Release(s)
Currently Existing in the System    Available for Installation   /
Install?
_________________ ____________________

OBP 3.1.2 1996/03/28 17:08          OBP 3.1.5 1996/08/27 16:13
no
POST 3.1.4 1996/04/09 03:23         POST 3.1.5 1996/06/28 11:54
no
Type sa if you wish to select all available firmware releases for
installation. Type h for help, quit to exit, or cont to continue: <=
Enter sa to update all PROMs.
      Firmware Release(s)              Firmware Release(s)
Currently Existing in the System    Available for Installation   /
Install?
_________________ ____________________

OBP 3.1.2 1996/03/28 17:08         OBP 3.1.5 1996/08/27 16:13      YES
POST 3.1.4 1996/04/09 03:23        POST 3.1.5 1996/06/28 11:54     YES
Type sa if you wish to select all available firmware releases for
installation. Type h for help, quit to exit, or cont to continue: <=
Enter cont at this point.
The Flash programming process is about to begin.
Type h for help, q to quit, Return or Enter to continue: <= Once
again hit Enter.
Erasing the top half of the Flash PROM.
Programming OBP into the top half of the Flash PROM.
Verifying OBP in the top half of the Flash PROM.
Erasing the bottom half of the Flash PROM.
Programming OBP into the bottom half of Flash PROM.
Verifying OBP in the bottom half of the Flash PROM.
Erasing the top half of the Flash PROM.
Programming POST into the top half of Flash PROM.
Verifying POST in the top half of the Flash PROM.
Programming was successful.
Resetting
```

7. At this point, physically power off the system. Then power it back on and let it come back up. Your programming is complete!

NVRAM—The Heart of the System

The nonvolatile random access memory (NVRAM) problems are among the most annoying. The contents of the NVRAM chip can become corrupted for a variety of reasons, most commonly: failure of the embedded battery. The battery embedded in the NVRAM chip keeps the clock running when the machine is off and also maintains important system configuration information. This section tells you how to reprogram your NVRAM chip and where to buy a new one, should you need to replace your current NVRAM chip. If your NVRAM is no longer functional, you'll need to purchase a new chip. For some problems, it is possible to reprogram your NVRAM, as long as your system retains its hostid and MAC address and the clock keeps time when the machine is turned off.

NOTE Sun Fire servers do not use NVRAM chips. Instead, NVRAM data is stored in the motherboard's SEEPROM (serial programmable read-only memory) in the workgroup servers and on the system controller boards' SEEPROM on the midrange and high-end servers.

NVRAM problems can usually be easily identified. Your system looks like it is trying to come up, but, because of its identity crisis, cannot do so. The output when the system is powered up will usually show an Ethernet address of all zeroes or all ones. Sun erroneously calls the media access control (MAC) address the "Ethernet" address. For those with Asynchronous Transfer Mode (ATM), Fiber Distributed Data Interface (FDDI), Token Ring, or other network cards, this is not a correct term; however to keep things simple and sane, we will continue with this mistake.

Here is an example of what one would see in the case of an NVRAM failure:

```
Sun Workstation, Model Sun-XXXXXX Series.
ROM Rev X.X, XXMB memory installed
ID PROM invalid.
Testing 0 Megabytes of Memory ... Completed.
ERROR: missing or invalid ID prom
Requesting Internet address for 0:0:0:0:0:0
```

or

```
Sun Workstation, Model Sun-XXXX Series.
Type 4 Keyboard
ROM Rev X.X, XXMB memory installed, Serial #16777215
Ethernet address ff:ff:ff:ff:ff:ff, Host ID ffffffff
Invalid format type in NVRAM
The IDPROM contents are invalid
```

How do you fix this? The solution is to replace the NVRAM chip. The NVRAM chip holds the hostid, MAC address, and most of the system boot parameters. When this chip fails, replacement is usually required. Another possibility is that the onboard battery has failed. If replacing the NVRAM chip fails, the battery or system board may need to be replaced, depending on whether the battery is soldered onto the system board. This is an extremely common error on older Sparcstation 1, 2, IPC, and IPX devices. The information in this section applies to the following Sun architectures: sun4c, sun4m, sun4d, sun4u, and sun3x (but not to sun4 and sun3).

Examples of Restoring the NVRAM

As we said earlier in this chapter, you should be careful that the first byte of the hostid matches the system type. Following are a few examples of modifying the hostid and Ethernet address.

NOTE Further information on NVRAM programming can be found at www.squirrel.com.

Example 1

To modify the hostid of an IPX to be 57c0ffee and the Ethernet address to be 08:00:20:c0:ff:ee, at the OpenBoot PROM monitor prompt, enter all of the following commands:

```
01 0 mkp
57 1 mkp
08 2 mkp
0  3 mkp
20 4 mkp
```

```
c0 5 mkp
ff 6 mkp
ee 7 mkp
57 8 mkp
0  9 mkp
0  a mkp
0  b mkp
c0 c mkp
ff d mkp
ee e mkp
29 f mkp
```

Note the simplification in this example. If you make the Ethernet address 08:00:20:H1:H2:H3 and the 4 bytes of the hostid ST, H1, H2, H3, where ST is the system type byte, and you put ST, 0, 0, 0 in the date-of-manufacture field, then the IDPROM checksum will always be 29 (remember, all of these numbers are hexadecimal). This makes things a bit easier. You can, in general, just enter the following:

```
01 0 mkp
real-machine-type 1 mkp
08 2 mkp
0  3 mkp
20 4 mkp
H1 5 mkp
H2 6 mkp
H3 7 mkp
real-machine-type 8 mkp
0  9 mkp
0  a mkp
0  b mkp
H1 c mkp
H2 d mkp
H3 e mkp
29 f mkp
```

And you don't need to calculate the checksum; it will always be 0x29.

Example 2

To change the hostid of an SS10 to 72c0ffee and the Ethernet address to 08:00:20:c0:ff:ee, enter the commands shown here:

```
01 0 mkp
72 1 mkp
08 2 mkp
0  3 mkp
20 4 mkp
c0 5 mkp
ff 6 mkp
ee 7 mkp
```

```
0  8 mkp
0  9 mkp
0  a mkp
0  b mkp
c0 c mkp
ff d mkp
ee e mkp
0 f 0 do i idprom@ xor loop f mkp
```

Example 3

To change the hostid of an SS1000 to 80c0ffee but leave the Ethernet address and the date of manufacture intact, enter these commands:

```
c0 c mkp
ff d mkp
ee e mkp
0 f 0 do i idprom@ xor loop f mkp
update-system-idprom
```

Note that the system type byte for the SS1000 is 0x80.

Example 4

To install a new NVRAM in an IPX and set the hostid to 57c0ffee and the Ethernet address to be 08:00:20:c0:ff:ee, do the following:

1. Turn the machine off.
2. Remove the old NVRAM chip.
3. Install the new NVRAM chip. Be sure to get the orientation right.
4. Turn the machine on.
5. At the OpenBoot monitor prompt, execute the following commands:

   ```
   set-defaults
   setenv diag-switch? false
   8 0 20 c0 ff ee c0ffee mkpl
   ```

6. Press Ctrl+D and then Ctrl+R.

The hostid on Solaris 2.5 x86 and Up

Intel processor machines don't have an IDPROM. Sun uses a different mechanism to generate the hostid. When the operating system is initially installed, a pseudo-random hostid is generated. It appears that this pseudo-randomly-generated hostid will always be between 1 and 3b9aca00. The hostid is based on 8 bytes of serialization information in the kernel module `/kernel/misc/sysinit`. This is in contrast to the situation on SPARC machines, where the hostid is based on the IDPROM.

The file /kernel/misc/sysinit contains code that initializes the variable hw_serial in the kernel based on the serialization information. On both SPARC and x86 versions of Solaris 2.5, hw_serial stores the hostid as a decimal C string.

Other than the 8 bytes of serialization information, the /kernel/misc/sysinit files do not differ between machines. Four of the serialization bytes depend upon the other four bytes, so the hostid is somewhat tamper resistant. If the serialization information is tampered with carelessly or if the sysinit module fails to load for some other reason, the hostid of the machine will be 0. A little more obfuscation is done in the code; hw_serial is not referenced directly in the module, but indirectly via the pointer _hs1107.

This means that, if you need to have two machines with the same hostid for some reason (say, to have a backup server with the same hostid in case your primary server malfunctions), you can simply copy the /kernel/misc/sysinit file from one machine to another.

Moreover, it seems that initializing hw_serial is the only function performed by the sysinit module. Hence, it is a simple matter to replace /kernel/misc/sysinit, yielding a machine with whatever hostid one wants by compiling a simple C program for a loadable kernel module that sets hw_serial to the desired value.

C code for a generic replacement sysinit module is included in change sun hostid, which is available from www.squirrel.com/squirrel/sun-stuff in the file change-sun-hostid.tar.gz. Replacing part of the operating system is probably not the best way to achieve this effect. In general, we recommend using one of the other modules to change the Sun hostid, because there is less risk of damaging things and rendering the system unbootable, but a few people have asked for this.

Summary

Hardware problems are almost always painful. Even if you never do any hardware maintenance yourself, knowledge of the symptoms of various types of failure and the commands available for bringing a system back to operational status will help you determine what's wrong and make sure that problems are resolved.

Hopefully you've found something relevant to your systems in this chapter that it will come in handy if ever a system refuses to boot. Systems are changing all the time and books like this one are not published more than once every few years. Take advantage of the information available to you on the Sun and SunSolve sites for updates of critical hardware information.

The most common reasons systems don't boot are because file systems are corrupt or there is a problem with the boot block on the disk. Less frequent problems, like those mentioned in this chapter, can occur, however, and you'll be ahead of the game if you know what other things can go wrong.

CHAPTER

16

Peripheral Vision: Understanding and Configuring Other Hardware

Probably one of the most frustrating things that we system administrators have to do is configure and troubleshoot peripherals. Back in the old days (SunOS 4.x, circa 1990), this task involved simply editing a few files and creating a few device files. The syntax of the old `/etc/printcap` files was a bit tedious, but the process of adding printers and updating the print service was simple once you got the hang of it. Then came Solaris with GUI administration tools and other ease-of-use "enhancements." Settings were managed through a pretty interface and we didn't *have* to understand how printing really worked. At first, we thought to ourselves, "Okay!" but our complacency lasted only until the first printing or connectivity problem. At that point, we missed having printer settings right there where we could see them.

Back in those days, many things were simpler. The Unix network didn't have to concern itself with printing to the PC network, and most modem connections supported only remote administration or simple Unix-to-Unix copy program (UUCP) file transfers. How times have changed! Now, networks are becoming dramatically more heterogeneous, and network-based services have taken over the role that host-based configuration files once played.

The setup and configuration of printers has also changed since the first edition of this book. The suggested tool for managing printers is now the Solaris Print Manager, and the choices for distributing information about printers has been expanded to include LDAP.

In this chapter, we highlight some of the changes in printer management that have come about in Solaris 9, detail the installation and configuration of both printers and modems, then show some handy tips and scripts for making these installations easier.

Managing Your Printers

Managing printers is quite a bit like managing children; just when you are certain that they understand your instructions to the letter, they run off and do something else entirely. Setting up printers through a GUI-based tool almost guarantees that at some point, something will not print correctly.

Under SunOS, we had it easy. We had a single printcap file that was NFS-mounted to all machines and then linked to `/etc/printcap`. A midnight `cron` job ran a script that created the spool directories for printers, added to it the printcap file of the day, and then restarted `lpd`. The simplicity and reliability of administrative solutions such as these made each of our lives as sysadmins quite pleasant. Just for historical purposes, we added comments to our printcap file (which, in time, contained several hundred printers), describing the name and the location of each printer. They looked like this:

```
#@  ws23045a    East BLDG    Column-3J    East Nowhere CT ttyb on
ws23045
```

Back then, Evan's organization owned a class B network, and all workstations and printers were named after the IP addresses they were using. The Dynamic Host Configuration Protocol (DHCP) had not yet entered the picture. If a printer was attached to a workstation, it was given the workstation's name with the port appended to the address. If it was a network printer, it was given the name pr######. For some strange reason, leading zeroes were dropped.

The `makespool` script lived under `/usr/remote/bin`, which was automounted to all systems. The script read as follows:

```
#! /bin/csh -f
#
# Make spool directories and restart printers
#
foreach printer (`grep "#@ " /etc/printcap | awk '{print $2}'`)
if ( ! -e /var/spool/$printer) then
    mkdir /var/spool/$printer
    lpc restart $printer
endif
```

Note how simple a script this is. Yet, it agreeably updated several thousand workstations nightly. But let's get back to reality—and today.

Solaris Printing

You have probably already added printers to your system using `admintool` or Print Manager. If you're a Unix command-line person at heart (like us), you most likely want to know what the tool is actually doing, and how the process could be scripted. This was certainly one of our first priorities on using the new print system.

The heart of the Solaris print engine is the `lpadmin` command. This command is used to add, remove, and change printers; to set the system's default printer; and to configure the `lp` print service. Our friend the `/etc/printcap` file is gone, and now configurations are saved in the `/etc/lp/printers/printername/configuration` files.

Here's an example of a local printer configuration file:

```
Banner on
Content types: postscript
Device: /dev/bpp0
Interface: /usr/lib/model/standard
Printer type: PS
Modules: default
```

For Solaris 2.6 and above, remote entries will look like the following:

```
Banner: on
Content types: postscript
Device: /dev/null
Interface: /usr/lib/lp/model/netstandard
Printer type: PS
Modules:
Options: protocol=bsd,dest=hp
```

Note the difference in the device lines and the addition of the `Options` line for the remote printer.

To configure a new printer, the -p parameter is used with `lpadmin`. To create a new printer from the command prompt, enter a command of the following form:

```
# lpadmin -p printername  [-v device | -s system!printername | -U
dialup] [-e printer_interface_to_copy | -.i interface | -m model]
```

The -v parameter is for the name of the device to which a local printer is attached. The entire path must be included in the device name for the printer to function. Here's an example—adding a local printer called *mickey* via command line to parallel port 0:

```
lpadmin -p mickey -v /dev/bpp0
chmod 600 /dev/bpp0
chown lp:lp /dev/bpp0
accept mickey
enable mickey
```

The -s parameter refers to the name of the remote system where the printer is physically attached. If the remote printer name differs from the local name, a ! separator is appended to the remote printer name. For example, if the printer hplaser is known on the system to which it is connected (say, sparc10) as printer `hp`, the command will read as follows:

```
lpadmin -p hplaser -s sparc10!hp
```

The -T parameter is used to define what type of input this printer can accept as defined in the /etc/termcap file. To have a complete remote install for the remote printer hplaser, which in this case we are saying is a PostScript printer, we would need to issue the following command:

```
lpadmin -p hplaser -T postscript -I any
```

For versions of Solaris prior to 2.6, the system_name (sparc10) also needs to be identified in the systems table located in the /etc/lp/Systems file. This is accomplished with the lpsystem command, removed in Solaris 2.7.

```
lpsystem -type bsd
```

The -U parameter tells the LP print service that the printer is connected remotely through a dial-up device, such as a modem. This parameter is almost never used.

To remove a printer, use the `lpadmin -x printername` command.

The -c class parameter places the printer into the specified class and creates the class if it doesn't already exist.

The -e printer name parameter copies the interface program of the printer specified as the interface for the printer.

The -i interface parameter creates a new interface program for the printer. The full path to the interface should be specified.

The -m model parameter selects the model interface program for the printer provided by the lp print service.

The -o options parameter shows the options that should be set in the printers.conf file.

The lpadmin command gives you considerable flexibility for defining printer features, filters, and alerts. The -D parameter allows the administrator to add a comment about the printer, such as its location, or the data types handled. This comment will be displayed by the *lpstat* command. The -A parameter allows you to specify the type of alert the LP print service should send when it detects a printing error. By default, the print service is configured to send email to root for every error encountered.

Another added feature is the ability to set up forms for printers. The default is no forms. Forms can be added with the `lpadmin -f allow:formlist` command. Form definition is handled by the lpforms command.

The printer can be made the default for a machine with the `lpadmin -d` command.

You can also enable or disable banner pages with lpadmin. However, we recommend that you edit the /etc/lp/interfaces/printername file instead. There is a line in the file that contains nobanner="no". This can be changed to "yes".

The traffic cop for the print services under Solaris is the lpsched daemon. This daemon handles multiple functions, including scheduling local and network print requests, applying filters when needed, executing programs required to interface with the printers, tracking forms and job status, and delivering all alerts. The lpsched daemon does not accept any commands directly. Instead, other lp print service commands communicate with it. These commands are shown in Table 16.1.

Table 16.1 Print Commands

COMMAND	ACTION
accept and reject	Accepts or rejects print requests
enable/disable	Enables or disables specified printers
lpadmin	Defines printers and devices
lpfilter	Adds, changes, deletes, and lists filters used with the LP print service
lpforms	Administers the use of preprinted forms with the LP print service
lpmove	Moves print requests from one destination to another
lpshut	Stops the LP print service
lpsystem	Registers remote systems with the LP print service; not needed in 2.6 and higher
lpuser	Sets printing queue priorities
lpsched	Operates on files that are located within the /etc/lp tree

The `lp` system files (contents of `/etc/lp`) shown in Table 16.2 are used by `lpsched` to clarify information from a print request before printing.

Table 16.2 lp System Files

FILE	CONTENTS
classes	Files identifying classes provided by the lpadmin -c command
fd	Existing filter descriptions
filter.table	Print filter lookup table
forms	Print form files
interfaces	Printer interface program files
logs	LP print service logs (linked to /var/lp/logs)
model	Standard printer interface program
printers	Directories for local printer definitions
pwheels	Printwheel or cartridge file definitions

New Printing Options from Solaris 2.6 to Solaris 9

Starting with Solaris 2.6, many changes were made to the printing subsystem to make life easier for us. Unfortunately, this happened at the same time we were just getting used to the other new way of doing things.

One major thing to remember is that this subsystem is *not* System V. Any remote machine attempting to print to a 2.6 or higher machine needs to refer to it as type BSD.

For Solaris 2.6 and above, a file called the `printers.conf`, which seems to behave quite a bit like the old Berkeley Software Distribution (BSD) `printcap` file, was added. Here is a sample `printers.conf` file:

```
hplaser:\
     :bsdaddr=sparc10,hp,Solaris:
_default:\
     :use=hplaser
_all:\
     :use=hplaser
```

In order to set up a remote networked printer under 2.6 or higher, the commands would look like this:

```
     lpadmin -p hplaser -o protocol=bsd,dest=sparc10!hp -T PS -I
postscript -v /dev/null -i /usr/lib/lp/model/netstandard
     chmod 666 /dev/null
     chown root:sys  /dev/null
     accept hplaser
     enable hplaser
```

A new command, `lpset`, can be used to add printers to the `printers.conf` file. The syntax is as follows:

```
lpset -n fns -a bsdaddr=server,printername,Solaris printername
```

If you are not using the federated naming system, the `-n fns` can be ignored.

The printer can even be aliased with new names by using the `lpadmin -p printername -c aliasname` command. Then issue the command `accept aliasname ans`, and that alias can now be used as well.

You can remove a printer with the new `lpset` command if it was added in a similar fashion. For example, the command

```
lpset -x printername
```

will remove the printer. The `lpget list` command will list all of the printers in your `printers.conf` file.

One nice thing about the `printers.conf` file is that it can live in the NIS and the NIS+ world and could be made available via LDAP as well. In fact, a map called `printers.conf.byname` can be configured into NIS for ease of sharing. Also, the master list can be pared down by the user, since users can now have their own `printers.conf` files located under `$home/.printers`. This file has the following format:

```
alias<tab>printer
_default<tab>printer
_all<tab>printer1,printer2,printer3
```

Note that *_default* becomes the default printer for the particular user, while *_all* is the list of printers that will be accessed via the `lpstat` or `cancel` command.

In Solaris 8, the recommended method for administering printers became Solaris Print Manager, a tool previously available with Solstice AdminSuite, but not included in the base operating environment. Print Manager is a Java-based tool with a graphical interface that is used to manage local and remote printers configuration (See Figure 16.1).

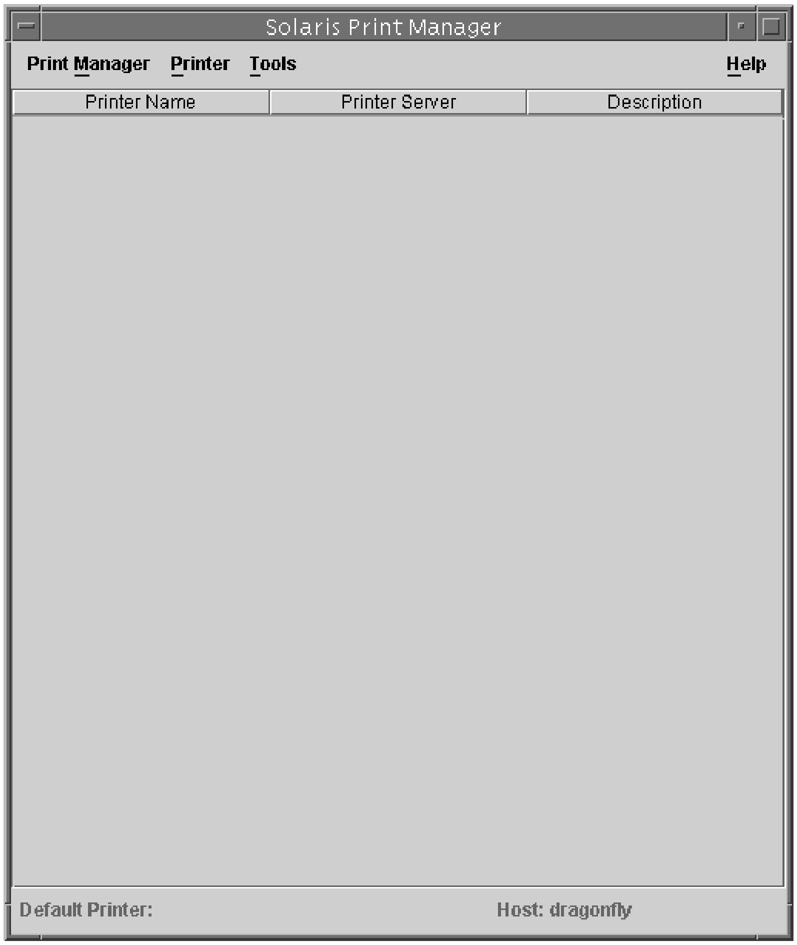
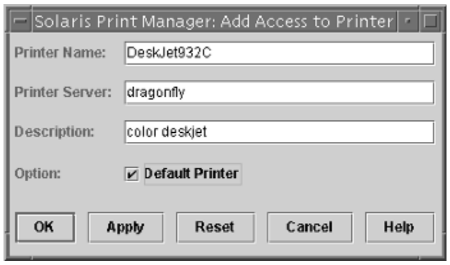


Figure 16.1: Print Manager.



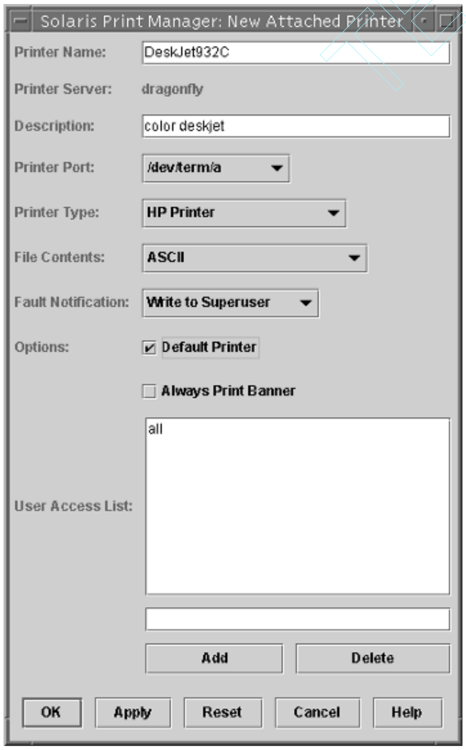
Solaris Print Manager: Add Access to Printer

Printer Name:	DeskJet932C
Printer Server:	dragonfly
Description:	color deskjet
Option:	✓ Default Printer

Figure 16.2 Adding access to a remote printer.

The forms used to add printers in Print Manager are not altogether different from the forms used in `admintool`. Figures 16.2 and 16.3 display forms being used to add a remote printer and a local (attached) printer, respectively.

Prior to Solaris 9, `lpsched` was started at boot time and ran continuously whether or not local printers were defined. In Solaris 9, `lpsched` is started by `lpadmin` when local printers are added to the system and then as long as these printers are defined.



Solaris Print Manager: New Attached Printer

Printer Name:	DeskJet932C
Printer Server:	dragonfly
Description:	color deskjet
Printer Port:	/dev/term/a
Printer Type:	HP Printer
File Contents:	ASCII
Fault Notification:	Write to Superuser
Options:	✓ Default Printer
	☐ Always Print Banner
User Access List:	all

Figure 16.3 Adding a local (attached) printer.

Solaris 9 also adds support for local printers connected through USB ports. The naming convention for these printers is `/dev/printers/[0 ... N]` (e.g., `/dev/printers/0`). Recommended PostScript printers are included in the man page for `usbprn`.

Solaris 9 also makes it possible to manage information about printers using LDAP. The Sun *System Administration Guide: Advanced Administration* (Sun 2000) book provides more information on using LDAP to support printers.

Tips and Tricks for Solaris 2.x Printing

To create a `printers.conf` file from the print system on a system prior to Solaris 2.5.1, use the following command:

```
conv_lp -d / -f /etc/printers.conf
```

If you have diskless clients, change the -d parameter to reflect the root directory for that client, and the -f parameter should reflect where the `printers.conf` file lives.

One problem that has plagued all of us is the watched kettle problem. How many times have we sent prints to a PostScript printer and watched the light blink, only to get no output? This is one of the most frustrating problems we've seen. This problem is usually caused by one of two things. Either the output type setting is incorrect, or the serial port setting is incorrect.

TIP **Sun provides a program located in `/usr/lib/lp/postscript` called `postprint`, for PostScript printing of ASCII files. This program resides in `/usr/lib/lp/postscript`. It translates text files into PostScript and adds formatting with the following operations:**

-c #	**Print # copies of each page. By default, only one copy is printed.**
-f *font_name*	**Print files using *font_name*. The default is courier, but any PostScript font can be used.**
-l #	**Set the length of a page to # lines. This is defaulted to 66 but will set based upon font and point size if set to zero.**
-n #	**Print # logical pages on each piece of paper, this is defaulted to one.**
-o list	**Print pages whose numbers are given in the comma-separated list.**
-r #	**Selects carriage return behavior. 0 is ignore, 1 is carriage return, 2 is Carriage Return/Newline.**
-s #	**Print files using point size #. Use with -l 0 to automatically scale lines per page.**

It is not widely known that PostScript printers attached to Solaris systems must have the type set to either postscript or simple (ASCII). The printing subsystem cannot autodetect PostScript and will most likely send incorrect data to the printer. For printers that autosense, setting the type as shown here will allow both PostScript and ASCII to print without a problem:

```
lpadmin -p printername -T PS -I postscript
```

Or, for HP LaserJet printers:

```
lpadmin -p printername -T hplaser -I simple
```

TIP Printing an ASCII Text file to an HP LaserJet or compatible printer from a Solaris machine may cause a strange effect whereby each succeeding line will be indented:

```
Line 1
     Line2
            Line3
                   ad infinitum
```

To fix this, edit `/etc/lp/printers/printername/configuration` and make sure it has the correct content type and printer type defined. For example, these two lines would define a PostScript HP LaserJet:

```
Content types: postscript
Printer type: hplaserjet
```

For settings on the serial ports, see the serial port section of this chapter. One EEPROM parameter should also be checked with the EEPROM command: `tty[a|b]-ignore-cd=true`. If the setting is false, the serial port will not transmit the data to the printer. If you reset this to true, a reboot will be needed for it to take effect.

How to test the serial port is a common question. First, make sure that you are using a null-modem cable. A null-modem cable crosses the send and receive wires (pins 2 and 3). If the baud rate is matched between the system and the printer, you should be able to `cat` a file directly to the port. For example, `cat mytestfile > /dev/cua/[a|b]`. Then, `echo "^D" >> /dev/cua/[a|b]`. The Ctrl-D character forces an end-of-job to the printer, and the file should then print. This little trick utilizes none of the print subsystem, but it verifies that your settings are correct. If printing works here, but not when you use the `lp` command, you should check that the baud rate is correctly stated in the printer definition. The baud rate can be changed by issuing the following command:

```
# lpadmin -p printername -o "stty=19200
```

Or, for a HP PCL plotter:

```
# lpadmin -p plottername -o "stty='19200 -opost'"
```

TIP Testing a printer by `cat`'ing a file to it with a proven null-modem cable is a good way to be sure your printer and port are working properly. Any ensuing problems would clearly be related to your print system and not these components. Don't forget the Ctrl-D (^D) character as explained in the text. This only applies to printers connected to serial ports.

Another common problem is when root is the only user that can print. This is usually caused by a permissions problem with /dev/null. The file should be 666 and should be owned by root:sys. This can be accomplished by issuing the following two commands:

```
chown root:sys /dev/null
chmod 666 /dev/null
```

TIP The `root:sys` argument changes both the owner and group of a file in a single command. This is a useful shortcut.

The last issue we tackle with respect to printers is the situation in which local printing works fine, but remote systems cannot print. This gets a little tricky. First, we must verify that the tcp listeners are still functioning. To do this, try the following:

```
# pmadm -l -p tcp
PMTAG          PMTYPE         SVCTAG          FLGS ID
<PMSPECIFIC>
tcp            listen         0              -    root
\x00020ACE000000000000000000000000 - c - /usr/lib/saf/nlps_server #
tcp            listen         lp               -    root    - - p -
/var/spool/lp/fifos/listenS5 #
tcp            listen         lpd            -    root
\x00020203000000000000000000000000 - p -
/var/spool/lp/fifos/listenBSD #
```

If these services are *not* running, set up a *local* printer on the print server with admintool. Here's the command line to set up the listeners:

```
# sacadm -a -p tcp -t listen       -c "/usr/lib/saf/listen tcp"
-v `nlsadmin   -V` -n 9999
#  pmadm -a -p tcp -s lp -i root -m `nlsadmin -o
/var/spool/lp/fifos/listenS5` -v `nlsadmin -V`
# pmadm -a -p tcp -s lpd -i root -m `nlsadmin -o
/var/spool/lp/fifos/listenBSD -A
```

```
'\x00020203000000000000000000000000'` -v `nlsadmin -V`
# pmadm -a -p tcp -s 0 -i root -m `nlsadmin -c
/usr/lib/saf/nlps_server -A
'\x00020ACE000000000000000000000000'` -v `nlsadmin -V`
```

Also, make sure that there is a process running that is listening on TCP port 515:

```
#  netstat -a | grep printer
*.printer            *.*                0      0     0      0  LISTEN
```

Hopefully, this brief guide will bring you to a better understanding of Solaris printing.

Serial Port Configuration (Terminals and Modems)

After printers, serial ports just *have* to be our favorite subject. Luckily for us, since the advent of the Internet, use of directly attached modem devices has shrunk considerably.

Those of us who come from the old SunOS 4.1 school remember how difficult setting up terminals and modems was back then. Editing the `/etc/ttytab` file and making sure the correct `getty` process was or wasn't forked took some amount of trial and error, but, sooner or later, our terminals and modems worked—and they worked well.

TIP Make sure you get the correct modem settings from your modem manufacturer. Incorrect modem settings usually cause more problems than system configuration.

With Solaris, our notions of the old regime went right out the window. `getty` processes were replaced with port monitors, and simple files to be edited were replaced by a slew of commands, all of which were difficult to remember.

For those of you just starting out, `admintool` is probably a good place to start. However, don't be fooled into believing that all of your serial port configuration can be performed via the admintool GUI. Knowing what happens under the hood will assist you in being able to determine whether you can use `admintool` and also whether it will work.

TIP Never try to use the Admintool GUI to set the baud rate for a serial printer. Always use the lpadmin command if you are not going to use the default of 9600 baud 8 bits no parity. The syntax is `lpadmin -p printername -o "stty=speed,character_set,parity"` where *speed* is the baud rate, *character_set* is either cs8 (8 bit) or cs7 (7 bit) and *parity* is either -parity (for no parity) or parity.

This section explains in detail how serial ports should be configured for both terminals and modems and explains the processes used to troubleshoot both types of devices.

Sun's serial ports come preconfigured with archaic settings. While most of the companies set their serial ports to 8 bits, no parity, and no stop bits, Sun's are preconfigured at 7 bits, even parity, with 1 stop bit. One of the first things that you will want to do is set the serial port modes to something useful.

Setting Serial Port Attributes

The first thing that you need to do is set the EEPROM settings for the port you are working with. The EEPROM commands still refer to the ports as `ttya` and `ttyb`; however, these translate into `/dev/term/a` and `/dev/term/b` in Solaris. To see the defaults, issue the following command:

```
#eeprom | grep tty
ttyb-rts-dtr-off=false
ttyb-ignore-cd=true
ttya-rts-dtr-off=false
ttya-ignore-cd=true
ttyb-mode=9600,8,n,1,-
ttya-mode=9600,8,n,1,-
```

To configure the ports as 34,800 baud with 8 bits, no parity, and 1 stop bit, issue the following commands. These settings require a reboot to activate.

```
#eeprom ttya-mode=38400,n,1,h
#eeprom ttyb-mode=38400,n,1,h
```

The /etc/ttydefs File

The `/etc/ttydefs` file contains all of the serial port mode definitions. The format of the entries in `ttydefs` is defined as follows:

```
ttylabel:initial-flags:final-flags:autobaud:nextlabel
```

For example:

```
contty:9600 hupcl opost onlcr:9600 sane::contty1
```

Following are some definitions for this example:

ttylabel. In the example, `contty` (`ttylabel`) is a port mode name, and is used as a parameter to the `ttyadm` part of the `pmadm` command when configuring a new port monitor. It is usually descriptive of the configuration parameters, such as 9600 or 38400E; however, the name has no bearing on the settings themselves.

initial-flags. This parameter contains the initial port settings taken from `termio` (`7I`). For example, the system administrator can specify what the default erase and kill characters will be. The initial-flags must be specified in the syntax recognized by the `stty` command. In the preceding example, the initial flags are

9600 baud; close the connection on last close (hupcl); map line feed to carriage return-linefeed (onlcr); and post process output (opost). There are a myriad of different settings, which can all be found in the man page for stty(1).

final-flags. The final-flags must be specified in the same format as initial-flags; ttymon sets these final settings after a connection request has been made and immediately prior to invoking a port's service. In the preceding example, the serial port is set to 9600 baud and reset (sane) to default values.

autobaud. If the autobaud field contains the character *A,* autobaud will be enabled. Otherwise, autobaud will be disabled. The ttymon process determines what line speed to set the port to by analyzing the carriage returns entered. If autobaud has been disabled, the hunt sequence is used for baud-rate determination.

TIP Don't use the autobaud field with modems supporting speeds higher than 9,600 baud, as it does not function with speeds in excess of 9,600. Using a combination of hitting the break key and a circular hunt group will perform the same function for speeds above 9,600!

nextlabel. If the user indicates that the current terminal setting is not appropriate by sending a <break>, ttymon searches for a ttydefs entry whose ttylabel field matches the nextlabel field. If a match is found, ttymon uses that field as its ttylabel field. A series of speeds is often linked together in this way into a loop called a *hunt sequence.* For example, 9600 may be linked to 4800, which in turn is linked to 2400, which is finally linked to 4800.

Note that the order of the flags within the initial-flags and final-flags fields is extremely important. The fields are processed from left to right. The sane command is extremely dangerous if used incorrectly. A long string of parameters ending with sane will reset the serial port to the default settings.

There are many ways to define the same thing. For example, evenp and -parodd parenb are the same. Both will set the port to even parity. There are many such possibilities.

The most common settings for use with modems and terminals are locked for sanity at 9,600 or 38,400 baud, 8 bits, no parity, hardware flow control. The entries in /etc/ttydefs would look like the following:

```
 9600L:9600 -parenb cs8 ignpar opost onlcr:9600 hupcl -clocal -parenb
-ignpar cs8 -istrip -ixon -ixany::9600L
38400L:38400 -parenb cs8 ignpar opost onlcr:38400 hupcl -clocal -parenb
-ignpar cs8 -istrip -ixon -ixany::38400L
```

Now that we understand the /etc/ttydefs entries and how they function, the next step is to delve into the world of port monitors.

Port Monitors and sacadm

As we mentioned previously, Solaris is quite different from its predecessor in its method of monitoring ports. Starting with Solaris 2, the notion of the Service Access Facility (SAF) was introduced, and the BSD `getty` method of port monitoring was abandoned. Now, instead of having *n* number of autonomous `gettys`, we have a single process (`/usr/lib/saf/sac`) controlling a group of independent port monitors. The tools used to administer these functions are the `sacadm` and `pmadm` commands. There can be multiple SACs (service access controllers) running, and each can control its own grouping of devices. For example, one SAC process can control a group of PPP-enabled modems, while another can control hardwired terminals.

If you have never dealt with the SAF before, initial setup of the port monitors might prove to be an exercise in futility. Because there isn't a simple set of files to edit, as with most Unix processes (and the `inittab` and `ttytab` in SunOS), all settings are initialized via the `sacadm` and `pmadm` commands. The steps required in order to configure a working service are as follows:

The `sacadm` command is used to add, remove, or modify SACs. Here is an example of a `sacadm` command that will add a SAC with a port monitor tag (`pmtag`) called *terminal.* The `pmtag` is actually the name by which this SAC is referenced. The type is `ttymon`; the command to run is `/usr/lib/saf/ttymon -v 1`.

```
sacadm -a -p terminal -t ttymon -c /usr/lib/saf/ttymon -v 1
```

The `sacadm` command can also be used to enable, disable, or remove the SAC with the -e, -d, and -r parameters.

To check to see what sacs are running, `sacadm -l` can be used. To show all running controllers of a specific type (e.g., `ttymon`), the command would be `sacadm -l -t ttymon`. To remove one controller, use the `sacadm -l -p {pmtag}` command. There is a default sac that runs called `zsmon`. To delete it and recreate it, use the following procedure:

```
#sacadm -r -p zsmon
```

To add a new one, use:

```
# sacadm -a -p zsmon -t ttymon -c /usr/lib/saf/ttymon -v `ttyadm -V`
```

At this point, no services should be running on this port monitor. This can be checked with the `pmadm` command. This command looks and acts very similar to the `sacadm` command, but operates on a lower level. Within the `pmadm` command is embedded the `ttyadm` command. The `ttyadm` command is used to format and display information specific to port monitors. The major parameters to both the `pmadm` and `ttyadm` commands are listed in Tables 16.3 and 16.4.

Table 16.3 pmadm Parameters

-a	Adds a port monitor.
-l	Lists the port monitors.
-r	Removes a port monitor.
-e	Enables a port monitor.
-d	Disables a port monitor.
-f u	Sets the *flag* field such that a utmp entry is created and a port monitor is enabled.
-p	Defines which SAC it should run under (pmtag reference).
-s	Identifies this port monitor under its SAC. This is called the *service tag* (svctag). You use this with the pmtag to specify this port monitor when you want to manipulate it.
-S y	Tells ttyadm to enable the *softcarrier* flag for this port. If an *n* follows the *-S,* it will be disabled, as for a modem-type port monitor.

Use the `pmadm` command to list the services running:

```
# pmadm -l
```

Check to see if there is already a service running on the port you wish to use. For this example, let's use `/dev/term/a`. In the following example, there is already a process running on `/dev/term/a` with the `svctag` of `ttya`. The `svctag` is equivalent to the `pmtag`, because it is the unique identifier for the `ttyadm` process on that port.

```
# pmadm -l
PMTAG          PMTYPE         SVCTAG         FLGS ID
<PMSPECIFIC>
zsmon          ttymon         ttya           u    root       /dev/term/a
I - /usr/
bin/login - 9600 ldterm,ttcompat ttya login:  - tvi925 y  #
zsmon          ttymon         ttyb           u    root       /dev/term/b
I - /usr/
bin/login - 9600 ldterm,ttcompat ttyb login:  - tvi925 y  #
```

Table 16.4 ttyadm Parameters

-d	The device file (tty node file) to run on
-l	The `/etc/ttydefs` entry to use
-s	The actual command to run

Next, we remove the existing service by issuing the following command with the `pmtag` and `svctag` found in the prior command:

```
# pmadm -r -p zsmon -s ttya
```

If there was no `pmtag` with the port we are configuring, we are all set. Because we already removed and added `zsmon`, there should currently be no services running.

Next, we need to configure the port for one of the following four types:

- Terminal use (`vt100/vt320/wyse`, etc.)
- Modem dial-in
- Modem dial-out
- Bidirectional modem

Configuring Ports for a Dumb Terminal

To configure the port for a terminal, first set up the port monitor based upon the terminal `sac` set up previously, as is done with this command:

```
pmadm -a -p terminal -s terma -i root -fu -v 1 -m "`ttyadm -S y -d
/dev/term/a -l 19200 -s /usr/bin/login`"
```

This command will set up a service that runs `/usr/bin/login` on port `/dev /term/a` at a baud rate of 19,200. The terminal should be configured to the same parameters as the port itself. In this case, we are running at no parity, 8 bits, and 1 stop bit (N/8/1).

Make sure you are using a null-modem cable. As mentioned earlier, a null-modem cable crosses the send and receive wires. Without a cable of this type, communication will not take place.

Configuring a Dial-in Modem

When configuring modems on serial ports, you *must* communicate with the modem via the `tip` or `cu` commands. These two commands allow you to test connectivity, and also allow you to configure the modem through the serial port.

For the `tip` and `cu` commands to be used, several things first need to happen. For one, the permissions on the port must be set such that you can read and write to it. This can be accomplished by changing the permissions on the appropriate device to 666 (rw-rw-rw-). The standard devices used are `/dev/term[a-z]` for incoming data and `/dev/cua/[a-z]` for outgoing data, although we have found that `/dev/term[a-z]` works fine for both. For another, the owner of `/dev/cua/[a-z]` should be `uucp:tty`, and `root:tty` should own the `/dev/term/[a-z]`.

The port monitor settings for a dial-in modem are similar to those for a terminal, with two exceptions. Primarily, the -S flag should be set to *n.* This ensures that there is a real carrier by changing the soft carrier flag to no. Also, the -b flag is set to allow bidirectional communication. Although we are configuring for dial-in only, the best way to test initial connectivity is with `tip`, and the -b flag is required for this testing. For a 38,400-baud connection, the `pmadm` line would look like this:

```
pmadm -a -p terminal -s terma -i root -fu -v 1 -m "`ttyadm -S n -d
/dev/term/a -l 38400 -s /usr/bin/login`"
```

`tip` utilizes a file called `/etc/remote` to determine its own settings. It contains information on how the system and the modem should communicate. Do not confuse this with modem-to-modem communication. Although the modem may be able to transfer at 56K per second, it is only possible to set the Sun modem speed to 38.4K per second. This speed is also called the *data terminal equipment (DTE) speed.*

Let's take a look at the `/etc/remote` file. There is a standard entry called `hardwire` that every administrator uses to test communications with modems. The default `hardwire` entry on Solaris looks like this:

```
hardwire:\
        :dv=/dev/term/a:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D:
```

Modify this entry to reflect the correct port-to-modem speed (i.e., 38,400) and outgoing device (`/dev/cua/a`). The resulting entry will look like this:

```
hardwire:\
        :dv=/dev/cua/a:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:
```

After this change, you can use the `tip` command to talk to the modem. The command is simply `tip hardwire`.

NOTE When you run this command, make sure you are in a nonscrollable window, because we have had problems with scroll bars in the past.

If everything is connected properly, you will see a *connected* message. At this point, you should be able to enter modem commands. If the modem understands, the industry-standard Hayes command `set ATZ` should return `ok`. If you see the `ok` returned, it is working. If you don't, you are not talking to the modem: Make sure that your cable is straight through and *not* a null-modem cable. Check that the baud-rate settings are correct, the modem is plugged in, and the `/etc/remote` and `pmtag` settings are correct. To exit this mode, type a tilde followed by a period (**~.**).

If you leave the port set as bidirectional and don't want others to dial out, change the ownership of `/bin/tip` to yourself, or make it not executable. If you want to change the port from bidirectional, you can either use `admintool` to unclick the bidirectional button, or delete and readd the `svctag` without the *-b.*

Configuring a Dial-out Modem

The interesting thing about configuring a modem for dialing out only is not what you need to do, rather it is what you don't need to do. For a dial-out-only modem, all the steps mentioned previously for dial-in modems should be followed to confirm communication with the modem. Once communication is verified, entries for your users to dial should be added to the `/etc/remote` file.

Although we briefly discussed this file in the preceding section, here is a more in-depth look at the `/etc/remote` file and the settings for dialing out.

The attributes that can be set in the file are shown in Table 16.5.

Setting up entries in `/etc/remote` for dialing out is fairly simple. For example, if you want to set up the phone number for Acme Widgets, 212-555-1234, using `/dev/cua/a`, you can create the following entry in `/etc/remote`:

```
acmewidget:pn=14155551234:tc=UNIX-38400:
```

Where `Unix-38400` looks like this:

```
UNIX-38400:\
          :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#38400:tc=modem1:
```

And `modem1` looks like this:

```
modem1:\
        :dv=/dev/cua/a:
```

When the users enter **tip acmewidget**, the phone number is looked up, the correct settings are applied, and the phone number 212-555-1234 is dialed via the Hayes command set (`atdt2125551234`).

Once connectivity is verified, the service running on the port that the modem is connected to should be removed via the `pmtag -r` command. This will prevent anyone from dialing into the system.

Table 16.5 /etc/remote Attributes

ATTRIBUTE	PURPOSE
`dv`	Device to use for the tty
`du`	Dial up; used to make a call flag
`pn`	Phone numbers
`at`	ACU type
`ie`	Input end-of-file marks (default is `NULL`)
`oe`	Output end-of-file string (default is `NULL`)
`cu`	Call unit (default is `dv`)
`br`	Baud rate (default is `300`)
`tc`	To continue a capability; point to another entry for the rest of the settings

Summary

Setting up printers and modems is one of our least favorite tasks. The command syntax is fussy, the settings have to be right on target for anything to work at all, and our users think we should be able to do this kind of stuff by twitching our noses. Yet every time we get a printer or modem working after struggling with it for a while, we feel like patting ourselves on the back.

The pointers and commands provided in this chapter should make the task a little less stressful and a little less error-prone.

CHAPTER

17

The Starfire

This chapter is intended as a relatively in-depth guide to Sun's early-generation domainable server, the Enterprise 10000. Although the newer generation of domainable systems is presented in the Sun Fire Chapter (Chapter 18), it is important to note that the two generations have little resemblance to one another. Many of the concepts that were originally presented in the Starfire have been updated and refined in the newer Sun Fire line, but Sun continues to purvey the Starfire as an alternative (and lower-cost) platform for consolidation. Of course, there are many eccentricities that make the Starfire particularly daunting at first glance, but in this chapter we hope to dispel any fears you may have and bring a coherent working picture of this *very* serious machine to light. We also would like to provide a solid foundation and reference to current owners and administrators of these systems.

What's a Starfire?

The original concept for Sun's Enterprise 10000 (or Starfire) was acquired from Cray Research. Despite the UltraSparc II heart and Solaris kernel, the E10000 was a new direction for Sun—to provide all the power and flexibility of a Unix system with the armor and reliability of a mainframe. What was forged was an utterly unique machine at the time, with a whole new set of ideals behind it.

Even from the outside, you can see that something is *different* about the Enterprise 10000. Physically, it is far larger than its Enterprise line cousins. Its striking lines were a significant departure from Sun's previous physical designs at the time of its creation. Under the hood, it's even more impressive. With up to 64 CPUs and 64 GB of memory, the E10000 (also called the *E10k*) is a performer through and through. Its layout is intelligent and eminently usable: 16 system boards divide up the whole E10000—each with the capacity for 4 CPUs (ranging from 250 to 400 MHz), 4 I/O cards, and 8 GB of memory. The system boards are connected through an intelligent high-speed backplane, which creates virtual links between the components for seamless symmetric multiprocessing (SMP). The backplane actually scales its allocated bandwidth to the size of the domain, allowing true flexibility, even on the fly. The system has one control board and two centerplane support boards (CSBs) that allow for system configuration and booting.

Although this machine is decidedly a Sun system, it differs from the rest of the Enterprise line in several noteworthy ways:

Partitioning. The E10000 can be segmented in up to sixteen separate logical machines. Each segment or *domain* is a fully functional Solaris system. *Ex unum, pluribus.* Each domain is also *completely isolated* from the next; thus a hardware or software error on one domain will never affect the others.

Dynamic reconfiguration. You can add or remove system boards on the fly from a running domain, allowing for greater flexibility of scaling. While the dynamic reconfiguration abilities of the newer Sun Fire systems have some significant improvements over the E10000, proper layout and domain design allow for reasonable DR capability.

Component-level blacklisting. If portions of the system board (including memory, CPU, I/O, etc.) are determined to be bad, they can be deallocated from the machine without physical removal or detriment to the system. The one stipulation is that this must be done before booting.

Floating network console. Domains have no physical console. Instead, they are fronted by the network console or *netcon*. This allows a system administrator to fix even the most serious problems without being physically present at the machine.

System service processor (SSP). The SSP is a front-end of sorts to the Starfire. It not only provides a front-line connection into the E10000, but also manages many of the features that make it so dynamic. SSPs are usually lower-end Sparcs, such as the Ultra 5. This is fairly analogous to the System Controller that has been implemented on the Sun Fire systems, but with a few more quirks and eccentricities.

Private network. A special network comprising only domains and the SSPs is mandatory with every E10000, providing unique new twists on network administration. However, it also provides a higher level of system accessibility (for sysadmins, at least).

Of course, having all of these features means having more administrative overhead, specifically in the arena of commands to remember. The intent of this chapter is to remove some of the mystique from this powerful machine and allow the reader to gain a clearer understanding of its more apocryphal features.

NOTE The use of *domain name* or *domain* in this chapter always refers to the name of a reconfigurable system partition on the Starfire, and never to DNS or other domain names.

Hardware Configuration

When you uncrate your E10000, there's a bit more work to be done than there is on your standard Enterprise machine, or its newer counterparts in the Sun Fire line. Planning is a necessity: You should know how you are going to configure your first domain or set of domains before the machine even arrives. Your domain is, of course, reconfigurable at any time, but planning will expedite machine setup immensely. The physical system is quite easy to get used to. There can be up to 16 system boards. Each side of the machine holds eight system boards and one control board, oriented vertically and arranged horizontally across the bottom of the system. You should have already discussed physical domain layouts on these boards with your Sun service rep, so we will gloss over this. Each domain must have not only the appropriate number of system boards (procs and memory) to support your application needs, but also the correct I/O on these boards to allow for all of the disk and peripherals you will need. Because each board has four sbus slots, you should have plenty of room, unless you are partitioning your machine into a multitude of one-board domains.

If dynamic reconfiguration is to be a goal on your E10K, a bit of work needs to be done to create a proper domain layout to support this feature. Because the Starfire uses system boards on which the I/O slots and CPU/memory components coexist, a primary board will always be present in your domain. This board cannot be dynamically reconfigured out of the running domain, because the kernel memory and boot disks will reside here. Although this is a limitation, additional boards within the same domain can be oriented such that the I/O layout will not impede the DR process. This is accomplished by dual-porting any relevant I/O (such as SCSI-attached disk, network connections, or Fibre Channel adapters) across multiple system boards and utilizing the multipathing features of Solaris to disable access to the impacted path during dynamic reconfiguration operations. While this will be discussed in more detail later, the best practice to keep in mind during system layout is that multiple physical paths to any I/O device is a necessity.

The absolute minimum configuration for any domain would be one Quad [Fast] Ethernet (QFE) card. An array or a boot disk is recommended, but one could technically boot a domain over the network. In regard to networks, you will need to use two ports on your QFE for each domain to correctly set up the Starfire. One should plug in to your public network (i.e., where the rest of your machines reside) and the other into a private network. Sun provides two small four-port Ethernet hubs for this purpose, but most users find them utterly inadequate (they are literally $49 off-the-shelf 10-baseT hubs). On our Starfires, we saw fit to replace them with 24-port 10/100 hubs. Though this may be overkill, the added potential capacity certainly will not be detrimental on a machine of this size. Plugged into this hub should be one Ethernet port from each domain, one from each control board, and one from the SSP. IP addresses will need to be allocated for the private network for domains and control boards. You should probably be consistent with regard to which QFE port you use on each domain, just for clarity's sake. The reason the control boards are plugged into the private network is for continuous access. This connection, called the JTAG, allows you to communicate with the host (albeit slowly) without any network drivers loaded or Ethernet interfaces configured or, indeed, without the system being up at all.

We have seen recommendations that suggest making the private SSP network the only network—actually plugging the private hub into your LAN. *This is an extremely bad idea.* The separation afforded by the private/public split will help you avoid problems in the future. Also, you probably would not want the SNMP traffic generated by the SSP software's monitors and such on your LAN. The best solution is to configure your SSP with two Ethernet interfaces, so that each domain and your SSP have both public and private interfaces.

With all of this planned out, the next step is to actually install the I/O where it belongs. It is generally best to have this done by a Sun-authorized service professional, because you probably wouldn't want to void the warranty on your system!

After the physical layout of your domains is complete, the next step is a logical configuration. Prior to beginning this step, you must name your domains. While the outside world will address your domain as it would any Solaris system (via DNS name), an important point should be made about domain naming conventions: Domain names as known by the SSP should decidedly differ from the actual hostname of the domain. So, if you plan the hostname of one domain to be *frood,* the name the SSP knows it as should *never* be *frood*. The reasoning behind this is simple: There are times when having both names the same could lead to confusion as to which Ethernet interface you are talking about on the domain, with potentially cataclysmic results. To prevent this, the simple convention we use is to add a *-dn* to the end of the domain name on the SSP, where *n* is simply the next number in a sequence. So, when we create the first domain, it would have *frood-d1* as a domain name on the SSP and *frood* as a hostname when we set up the OS.

Now you are almost ready to configure your domains. However, before this can happen, you will have to familiarize yourself with the SSP.

The System Service Provider—Intimidation at Its Finest

The control center of the E10K resides within its *system service processor* (SSP). This standalone machine acts as the software controller for most of the Starfire's unique features. In fact, much of this chapter is devoted to commands and concepts that reside on the SSP. While the System Controller that has been implemented in Sun Fire systems is an updated version of the same concept, the SSP has a bit more of a raw feel to it. The System Controller presents a streamlined interface to system access and dynamic reconfiguration commands, while the SSP makes it a bit more difficult to accomplish similar results. The machine generally has a standard Solaris build (up to Solaris 8 as of this writing), with the singular addition of the SSP packages. These are provided by

Sun on a separate CD. Installation is as simple as a `pkgadd` command and a few brief configuration questions after reboot, but using it is an entirely different matter.

When setting up the SSP, you will need to specify a *platform name.* This is the name by which the SSP will reference the actual E10000, and is used in many domain-configuration commands and tasks. For the sake of argument, let us use the example of two Starfires. On one, the platform name could be *osiris,* and on the other, *horus.* This is simply for clarity when referencing the system as a whole. The SSPs should be named accordingly: The hostname on the first might be *osiris-ssp01,* and on the other, *horus-ssp01.*

Another important note is that the SSP software setup will prompt you for whether the machine you are setting up is to be a main or a spare SSP. There are generally two SSPs shipped with each E10000. To state the bluntly obvious, one should be main, one spare. The purpose of this configuration is that, in theory, if you were to lose an SSP due to a crash, an act of the deity, or the use of a Tesla coil in the nearby vicinity, you could set the secondary up for use and continue without much problem. The actual functionality to support this feature comes in the SSP startup files, located in `/opt/SUNWssp/bin`.

The setup of the SSP v3.2 software is shown in the following listing. You will be prompted with this screen after you install the packages on the SSP CD-ROM and reboot. This process can also be manually invoked with the `ssp_install` script. In the following example, the control board IP addresses are not listed in `/etc/hosts`. Though it is generally recommended that you do this (once you assign the IPs), it is not absolutely necessary. Obviously, the values here are fictitious. Your mileage may vary depending on the version of SSP you're installing, as well:

```
Beginning setup of this workstation to act as a MAIN or SPARE SSP.
The platform name identifies the entire host machine to
the SSP software. The platform name occupies a different
name space than domain names (hostnames of bootable systems).
Please enter the name of the platform this ssp will service: ouroboros
Do you have a control board 0? (y/n): y
Please enter the host name of the control board 0 [ouroboroscb0]:
I could not automatically determine the IP address of ouroboroscb0.
Please enter the IP address of ouroboroscb0: nnn.nnn.nnn.nnn
You should make sure that this host/IP address is set up properly
in the /etc/inet/hosts file or in your local name service system.
Do you have a control board 1? (y/n): y
Please enter the host name of the control board 1 [ouroboroscb1]:
I could not automatically determine the IP address of ouroboroscb1.
Please enter the IP address of ouroboroscb1: nnn.nnn.nnn.nnn
You should make sure that this host/IP address is set up properly
in the /etc/inet/hosts file or in your local name service system.
Please identify the primary control board.
Is Control Board 0 [ouroboroscb0] the primary? (y/n) y
Platform name   = ouroboros
Control Board 0 = ouroboroscb0 => nnn.nnn.nnn.nnn
Control Board 1 = ouroboroscb1 => nnn.nnn.nnn.nnn
Primary Control Board = ouroboroscb0
Is this correct? (y/n): y
Are you currently configuring the MAIN SSP? (y/n) y
MAIN SSP configuration completed
```

There are several directories on the SSP with which you will need to familiarize yourself, as you will undoubtedly be frequenting them. The actual SSP binaries and libraries are installed by default in `/opt/SUNWssp`. This can be referenced by the SSP user as the environment variable `$SSPOPT`. The `bin` directory under this hierarchy is included in your path as the user *ssp* by default. Most of the important commands for administering an E10000 (and the ever-important man pages to back them up) can be found here. A slightly more obscure but nonetheless important directory is `/var/opt/SUNWssp`. This can be referenced by a similar environment variable, `$SSPVAR`. Many of the actual configuration files for your specific E10000 can be found here, along with the domain-specific and SSP-specific message files and error logs. These are discussed in detail in the *Knowing Your SSP* section a bit later in the chapter.

Though the default environment for the SSP user is obfuscated, or even obnoxious to some, it is an unfortunate necessity. The intricate ways in which the SSP user's environment interfaces with the E10000 domain commands require some very specific environment variables, and, quite unfortunately, csh. It is not recommended that you alter the SSP user's environment in any more than superficial ways, such as the background color in OpenWindows, or the number of command tools that are brought up for netcons.

Logging in as the SSP User

The first thing you will have to adjust to when learning Starfire administration is *not* logging in as root. Instead, most of the important work can be done as the user *ssp.* This user is installed by default with the SUNWssp package. This account will allow you to execute all of the important commands that will be discussed in this chapter. When you log in as *ssp,* you will notice an immediate difference.

```
Login: ssp
Passwd:
Last login: Tue Jun 29 15:11:51 from 167.69.134.221
Sun Microsystems Inc.    SunOS 5.5.1    Generic May 1996
Please enter SUNW_HOSTNAME: foonly-d1
ouroboros-ssp01:foonly-d1%
```

Upon initial login as ssp, what you enter as a default domain (SUNW_HOSTNAME) is not important. At this point, there are no domains to communicate with. This will become important later—the domain you set is the one you will be talking to with Starfire commands. Upon initial setup, there are no domains created. The easiest way to determine domain configuration is with the command `domain_status`. The output of this command will look as thus upon initial setup:

```
ouroboros-ssp01:foonly-d1% domain_status
domain_status: No domains configured.
```

In order to configure a domain, you will need an EEPROM image. Because the E10000 is dynamically configurable, it made little sense to have physical EEPROM chips. Instead, the EEPROM images are stored as data in `$SSPVAR/SUNWssp/etc/platform_name`. This is generated from a template image in `$SSPVAR/.ssp_private/eeprom_save` by `domain_create`. Sun will provide you with one

EEPROM image for each domain you have requested. If a new EEPROM image needs to be generated, Sun can provide you with a key and hostid for use with the `sys_id` utility. You should make sure that you have a written copy of the initial Sun-provided hostid and serial number in case anything catastrophic should ever happen to the machine.

The next important concept to understand before a domain can be created is the software-controlled power feature of the Starfire. Each system board and control board can be powered on or off from the domain, and the status of each can be checked. The command to accomplish such feats is `power`. At its simplest usage, power is invoked like this:

```
power -on -sb 0
power -on -cb 0
```

These commands power on system board 0 and control board 0, respectively. Realistically, control boards do not need to be powered up or down often, only upon initial setup of the E10K. The antonym of the `power -on` command would result from replacing *-on* with *-off*. However, the `power` command is considerably more powerful than that. Used with no arguments, it will display the power status for each subsystem on each board. It can also be used to adjust acceptable voltage margins, turn off the entire machine, or bring the center plane up or down. If you substitute the *-all* option for the system board option, the command will affect all system boards and control boards.

WARNING It is generally advised that any functions besides the most mundane of cycling power on system boards be left to Sun service professionals.

Creating Domains

Before you can create your first domain, you need to make sure that all of the system boards you will be using are powered on. Once the system boards are available, the command to begin this process is `domain_create`. The syntax is as follows:

```
domain_create -d domain_name -b sysboards -o os_version -p platform
```

So, if your first domain is `foonly-d1`, the platform name you decided on is *ouroboros*, it will live on system boards 0 1 and 2, and you will be running Solaris 2.5.1, the command will look like this:

```
domain_create -d foonly-d1 -b 0 1 2 -o 2.5.1 -p ouroboros
Domain foonly-d1 created!
```

The same command can be used to create any future domain. Care should be taken (through use of the `domain_status` command) to make sure that the system boards that you wish to use in the creation of a new domain are not already in use by another.

```
Ouroboros-ssp01:foonly-d1% domain_status
DOMAIN          TYPE                       PLATFORM      OS     SYSBDS
foonly-d1       Ultra-Enterprise-10000     ouroboros      8     0 1 2
```

Also, it is worthwhile to note the somewhat obvious point that the boards you utilize for a domain must have the minimal configuration that any Sun machine would require: enough memory and CPU, an Ethernet interface, and so on.

Removing a Domain

Of course, because the E10K is dynamic, you may want to remove a domain at some point. This is very easy to do, and it can be done on a domain that is up and running, so be extremely careful when invoking the removal of a domain. Fortunately, the command is polite enough to prompt you before taking any action. The syntax is simple:

```
domain_remove -d domainname
```

You will then be asked if you want to keep subdirectories related to the domain. We generally recommend answering yes, because you never know when you might want to check old logfiles. Beyond that, the directories are keyed by name, so if you recreate the domain (which is often the case), you will have everything just as you left it. Here is an example:

```
domain_remove: The following subdirectories contain domain specific
information such as messages files, configuration files, and hpost dump
files.
You may choose to keep these directories if you still need this
information.
This domain may be recreated with or without this information being saved.
                /var/opt/SUNWssp/adm/foonly-d1
                /var/opt/SUNWssp/etc/ouroboros/foonly-d1
Keep directories (y/n)? y
Domain : foonly-d1 is removed !
```

The command `domain_history` will show you a complete history of domains that have been removed. Its output looks like a `domain_status`.

Knowing Your SSP

To become a true acolyte of the Starfire, you must learn the inner workings of the SSP software. To a great extent, this involves plying through the directories related to the SSP and looking at the log files, scripts, and configuration files therein. The SSP also runs various daemons to keep the state of the environment constant and monitored. Although in most cases it is not necessary to directly interact with any of these programs, having a working knowledge of them does indeed aid in administration.

For the most part, the salient bits of the SSP (besides the binaries themselves) are located in the directory referenced by the `$SSPVAR` environment variable when you are logged in as the ssp user. As discussed earlier, the default value for this is

`/var/opt/SUNWssp/`. The most directedly important subdirectory in this hierarchy is `adm`, which contains the SSP message file (similar to the system's `/var/adm/messages` but specific to the SSP software) and domain-specific message and log files.

The `$SSPVAR/adm/messages` file is, for the most part, relatively incomprehensible, and primarily contains messages from some of the daemons, which we will elucidate shortly. Occasionally, though, very helpful messages come through, dealing with the powering on or off of a system board, temperature warnings, or fan failures. These are generated by the event detector daemon. We have included a Perl script in the *A Helpful Script* section that can monitor multiple SSPs for such messages from a remote location. It can then perform an action, such as paging or emailing you.

The most important files are located in subdirectories of `$SSPVAR/adm`. Within this directory, there will be directories with the same names as your domains. Each of these contains domain-specific message files, records of each hpost run, and any error dumps you may incur. These become important after a crash or panic, because the information collected there can allow Sun (or you, if you know how to interpret core files) to determine the root cause of the problem. We discuss core files in the *Recovering from Crashes* section later in this chapter. For now, let's look deeper at the domain-specific files.

messages. A messages file for the domain from the SSP's point of view. It contains specific information about the interaction of SSP daemons and the domain. Most of these messages are generated by the event detector daemon.

post. Subdirectory of date/time-keyed files containing the output of each hpost run.

last_edd_bringup.out. The output of the last bringup run.

Edd-Arbtstop-Dump, **Edd-Record-Stop-Dump**, **hostresetdump.** These are dumps, discussed in the "Recovering from Crashes" section later in this chapter.

As for the rest of `$SSPVAR`, no other directories contain files that are particularly interpretable by mortals; most of the contents are state or configuration files for the various SSP daemons. The files located within these folders should *never under any circumstances* be edited, with the one exception of the blacklist. Changing these files could lead to system failures. Following are some brief descriptions, for the curious:

$SSPVAR/etc/*domainname*. Contains the blacklist file and domain-name subdirectories containing the EEPROM images.

$SSPVAR/pid. Contains the PIDs of various SSP daemons.

$SSPVAR/data/Ultra-Enterprise-10000. Contains the E10000 JTAG scan database, used by the `cbs` daemon.

$SSPVAR/.ssp_private. Contains lockfiles created by the file access daemon, and some resource files for the control board. Here also are several files containing domain and SSP configuration information, which are accessed by SSP commands. These include `domain_config`, `domain_history`, `ssp_resource`, and `ssp_to_domain_hosts`.

There is another SSP directory, `$SSPETC`. The default location for this is `/etc/opt/SUNWssp`. Within this directory resides the `ssp_startup` script. This starts the daemons that enable communication between the SSP and the domains. These should never be run manually. They are as follows:

cbs. Control board server. This provides a conduit to the control board for the rest of the SSP servers and clients. It also manages the JTAG functions.

edd. Event Detector daemon. This listens for events such as temperature warnings, fan failures, and panics, and creates SNMP traps to handle the response. It also creates dumps, such as arbstops or recordstops.

snmpd. Simple Network Message Protocol daemon. Handles the management information base (MIB) for the E10000.

straps. Monitors the SNMP traps and passes them to any SSP client software, such as EDD.

fad. File Access daemon. This provides file locking for client programs.

machine_server. Handles port lookup or allocation requests for `snmpd` and `netcon_server`.

obp_helper. Also reexecuted every time a bringup is run, this downloads the OpenBoot PROM to the domain's memory.

netcon_server. One for each domain, also reexecuted at bringup.

SSP Command Reference

You can find additional information about the Starfire along with a command reference at the Sun Web site http://docs.sun.com. This site provides online access to an expanding library of Sun's documentation. The site provides tools with which you can search or browse the collection. If you choose to browse, you can do so by subject, collection, or product. You can also search by OS (e.g., Solaris 8) or product classification (e.g., servers). Much of the site is organized as document collections. You'll find release notes as well as user guides and installation manuals. For the Starfire, you'll find user guides as well as reference manuals. The site also provides links to other documentation sites. We strongly encourage you to browse the material available here.

A Helpful Script

The script `sspmon.pl.` (see the following listing) will monitor the `$SSPVAR/adm/messages` on the listed SSPs. The assumption is that the machine you run this from is in the `.rhosts` file of the user you are running it as on the SSP. The reason it is set up for remote access is to monitor multiple SSPs at once. This will simplify monitoring and allow you to work from one primary location.

```
#!/usr/local/bin/perl
# Remotely monitors $SSPVAR/adm/messages on ssp. Run every 15 mins
# from cron
# There is absolutely no implied warranty. Use at your own risk!
# 8/26/98 Jeff Ruggeri
# Last modified 04/08/99, JRR
#Put the platform name (or any handy referential bit of info)
#and the SSP's IP in this hash table.
%hosts = ( "STARFIREone" => "127.0.0.1",
           "STARFIREtwo" => "127.0.0.2"
         );
#Action - the actual binary to run. This should take the message as an
#argument. Listed below is an example - it calls an external paging
#script. It would be very easy to mail as well.
$action = "page.ksh RUGGERI";
$date = '/usr/bin/date +"%b %e "';
$hour = '/usr/bin/date +"%H"';
$min = '/usr/bin/date +"%M"';
chomp($date, $hour, $min);
if ($min < 10) {
   $hour--; }
$mdate = $date . $hour;
if (-e "/tmp/.sspmon.state") {
   open (STATE, "/tmp/.sspmon.state");
   flock STATE, 2;
   @state = <STATE>;
   flock STATE, 8;
   close (STATE);
   unlink ("/tmp/.sspmon.state"); }
HOSTLOOP: foreach $host (keys %hosts) {
  @msgs = 'rsh $hosts{$host} cat /var/opt/SUNWssp/adm/messages | grep
\"$mdate\"';
  if (scalar @msgs == 0) {
     next HOSTLOOP; }
  else {
   @fail = grep /(FAILED)|(OFF)|(911\stemp)/, @msgs;
   if (scalar @fail == 0) {
      next HOSTLOOP; }
  foreach $grr (@fail) {
     chomp($grr);
     $grr =~ s/^.*\(.*\).*:\s//;
     $argh{$grr} = 1; }
  foreach $failu (sort keys %argh) {
     $message = "E10K Alert ($host): " . $failu;
     if (scalar @state) {
        foreach $old (@state) {
           chomp($old);
           if ($message eq $old) {
              $oldmess{$message} = 1; } } }
     if (!exists $oldmess{$message}) {
        system "$action \"$message\"";
        $oldmess{$message} = 1;
        print "$message\n";  } }
  } }
open (STATE, ">/tmp/.sspmon.state") || die "Can't open state file";
```

```
flock STATE, 2;
foreach $mess (keys %oldmess) {
   print STATE "$mess\n"; }
flock STATE, 8;
close (STATE);
exit 0;
```

bringup—Your Key to the E10000

Now you are undoubtedly at a loss. The domain is set up, so the machine is essentially *there*, but how does one access it? How can one even power it up, or run a POST? The answer lies in the `bringup` and `netcon` commands (`netcon` is discussed in the next section). While not as straightforward as the system controller commands on a Sun Fire, the usage is similar. The `bringup` command acts as a virtual system key; it lets you start your domain up and access the OpenBoot PROM, or bring it up to the OS if one exists. Before issuing this command, the status of the domain can be determined with `check_host`. This command will return a straightforward `Host is UP` or `Host is DOWN` for the domain currently set as `$SUNW_HOSTNAME`. It is generally not advisable to run `bringup` on a running domain unless it is an emergency or recovery situation.

This command has a very specific set of functions that it executes, in this order:

1. First, `bringup` runs `power` to check that all of the system boards in the domain are powered up. If not, it will abort with a message to this effect.
2. Next, it runs `check_host` to determine the status of the domain. If the domain is determined to be up, `bringup` will prompt you for whether or not to continue. This is useful if you are using this command to recover from a hung host.
3. The `blacklist` file, located in `$SSPVAR/etc/platform_name/blacklist`, is checked. This file allows components, from I/O units and CPUs to entire system boards, to be individually excluded from the domain at start time. This file is incredibly useful, and is described in the *Blacklisting for Fun and Profit* section later in this chapter.
4. Then, `bringup` will check the status of other domains on the system. If no other domains are active, it will prompt you with the fact that it intends to configure the centerplane and ask if you want to continue. If the centerplane is already configured, reconfiguring it will not cause any harm.
5. Next, `bringup` runs `hpost` on the domain. This is a very valuable tool that can be run (on a domain that is not up) interactively at any time. It runs the domain through a series of tests that can often shake out hardware errors. By default, it will run at level 16. It can be configured to run up to level 127, which we will discuss a bit later.
6. Finally, `bringup` starts the `obp_helper` and the `netcon_server`, both of which are final handoffs to the system itself.

The most important thing to keep in mind when running `bringup` is that the domain that is currently set as the `$SUNW_HOSTNAME` environment variable is the one the command will affect. This is true of most E10000-specific commands. This is the hostname that you enter upon logging in as SSP, and the one that will be prominently displayed in your prompt (PS1). It is important to *always* check what this value is set to before executing a domain-specific command. To switch between domains, use the command `domain_switch domainname` (pretty straightforward).

`bringup` can be run with many options, which are profiled at the end of this chapter. The most basic and important option is *-A*. A `bringup -A off` will bring the system to the OpenBoot PROM by default, whereas a `bringup -A on` will boot the system normally (that is, if an OS is installed in one way or another). You can also pass boot arguments to the OpenBoot PROM by specifying them at the end of the command line.

Following is the complete output from the successful completion of a `bringup` command. This is what the output should look like (with the exception of the configuration-specific lines) when `bringup` runs with no errors:

```
ouroboros-ssp01:foonly-d1% bringup -A off
This bringup will configure the Centerplane. Please confirm (y/n)? y
Starting: hpost  -C
Opening SNMP server library...
Significant contents of /export/home/ssp/.postrc:
#
logfile
phase cplane_isolate: CP domain cluster mask clear...
phase init_reset: Initial system resets...
phase jtag_integ: JTAG probe and integrity test...
phase mem_probe: Memory dimm probe...
phase iom_probe: I/O module type probe...
phase jtag_bbsram: JTAG basic test of bootbus sram...
phase proc1: Initial processor module tests...
phase pc/cic_reg: PC and CIC register tests...
phase dtag: CIC DTAG tests...
phase mem: MC register and memory tests...
phase io: I/O controller tests...
phase procmem2: Processor vs. memory II tests...
phase lbexit: Centerplane connection tests...
phase npb_mem: Non-Proc Board MC and memory tests...
phase npb_iopc: Non-Proc Board IOPC register tests...
phase npb_io: Non-Proc Board I/O controller tests...
phase npb_cplane: Non-Proc Board centerplane connection tests...
phase nmb_procmem2: Non-Mem Board Proc vs. memory II tests...
phase final_config: Final configuration...
Configuring in 3F, FOM = 215040.00: 20 procs, 7 Scards, 12288 MBytes.
Creating OBP handoff structures...
Configured in 3F with 20 processors, 7 Scards, 12288 MBytes memory.
Interconnect       frequency is 83.284 MHz, from SNMP MIB.
Processor external frequency is 124.926 MHz, from SNMP MIB.
Processor internal frequency is 249.851 MHz, from proc clk_mode probe.
Boot processor is 0.0 =0
```

```
POST (level=16, verbose=20, -C) execution time 6:53
Boot proc 0 is written to /var/opt/SUNWssp/etc/ouroboros/foonly-
d1/bootproc
Updating the bootproc and IntVector MIB
Starting: obp_helper -m 0  -A off
Starting: netcon_server -p 0
```

If the `hpost` phase of `bringup` does, in fact, uncover any hardware problems, `bringup` will exit immediately with the error encountered. If this occurs, it may be worthwhile to capture the information on the screen. Open a call with Sun immediately, as `hpost` failures are generally quite critical. You may also want to run the domain through a more thorough hpost, depending on Sun's assessment of the situation. This can be accomplished with an `hpost  -l127` command on the current `$SUNW_HOSTNAME`, but be forewarned that this can take many hours, depending on the size of the domain in question.

The default run level for `hpost`, within a `bringup` or otherwise, is 16. This runs basic tests on the proc, memory, and I/O devices. If you decide (due to a hardware failure, or simple paranoia) that this is not sufficient, the default level can be modified with the file `~ssp/.postrc` (`~ssp/` being SSP's homedir, wherever that may lie). This is a simple-format ASCII text file, which should contain the following line:

```
level n
```

where *n* is an integer 7 to 127. Detailed descriptions of the important levels can be displayed by running `hpost -?level`.

NOTE Many other options can be added to the `.postrc`, but *level* is the only one of real use. Most of the others are for debugging or Sun Service engineer use. Full descriptions can be found in the man page for `postrc`.

Once `bringup` completes, the system is ready to be accessed. The important question here is: How do you access it? This is a perfect segue to `netcon`.

netcon—It's Your Friend

`netcon` is an extremely versatile and convenient tool. In fact, we would list it as our favorite feature of the Starfire. `netcon` stands for *network console,* and it is just that: a free-floating console for your domain, accessible from the SSP. This functionality is extremely useful in two ways. First, there is no need to connect physical consoles or terminal server lines to each domain. This could be exasperating, depending on how much domain reconfiguration you do. Also, if problems occur with the system, a physical console presence is not necessary. All of the console functions that would generally require a sysadmin on-site can be accessed from anywhere, anytime, without the presence of terminal servers. This is a life-saver if a system goes down in the middle of the night. Instead of driving in to fix a problem, it can be done from a dial-up terminal, or anyplace from which SSP access can be gained. This functionality became so popular that Sun has implemented it across the entire Sun Fire line, providing access very similar to `netcon` functionality through its built-in System Controller cards. While `netcon` is not as clean an implementation as the system controller, it has a serious advantage over the Enterprise line that was current when the E10K was released in August 1998.

Before `netcon` will work, however, the domain must be brought up (see the previous section). If a `netcon` session is started when the host is down, it will sit idly and wait for the `netcon` server to be started on that domain. The process on the SSP that needs to be running in order for this scenario to work is `netcon_server -p boot_proc`. This will be started automatically by `bringup`. The process that runs on the domain (once at the OS level) is the *cvc daemon* (`cvcd`). `cvcd` allows the domain to communicate with the SSP over the private Starfire network interface. Prior to `cvcd` being brought up, all communications are transacted over the JTAG. The JTAG is the Ethernet interface on the control board, which is also plugged into the private Starfire network. This is generally painfully slow, but it grants the access necessary to bring the system up. This is an important distinction from the methodology employed on the Sun Fire systems, which use only backplane communication to the domains, with no private networking or special daemons. It should be noted that if you are connected with `netcon` in multiuser mode, and the system is still responding over the JTAG (which is immediately noticeable; it's *that* slow), you can manually restart the `cvcd`. You should probably kill the previous one if it is still running, then simply run `/sbin/cvcd`.

System Installation

For an initial system installation, follow the normal procedures that you would for a JumpStart boot from the OpenBoot PROM (see Chapter 6). The only notable difference between this and any other Enterprise system is that the install will always be in character mode, so just be sure to set your terminal type correctly when prompted. From the SSP, assuming you are using `cmdtool` or something similar, X-Terminal usually works well.

So, if you run a `netcon` immediately after running a `bringup -A off`, you might see this:

```
ouroboros-ssp01:foonly-d1% netcon
trying to connect...
connected.
SUNW,Ultra-Enterprise-10000, using Network Console
OpenBoot 3.2.4, 12288 MB memory installed, Serial #00000000.
Ethernet address 0:0:be:a6:00:00, Host ID: 00000000.
<#0> ok
```

Hey, that looks familiar! So, you're up and running. From here on in, you can address the system much as you would any Enterprise server.

Connecting with netcon

Connecting with `netcon` is as simple as issuing the command. Once connected, your terminal becomes a virtual console for the current working domain. You can interact with the system at the OpenBoot PROM either in single-user mode or multiuser mode. However, keep in mind that until `cvcd` is running in multiuser mode on the domain, all communication will be over the JTAG. Now, because `netcon` is a generally accessible system tool, it follows that multiple users can run it at once. Confusion is alleviated by allowing only one user at a time to have write access, but an indefinite number to have read access. There are several modes in which a `netcon` window can be opened: *locked write* permission, *unlocked write* permission, and *read-only* permission. Unlocked write permission is not particularly secure, because it can be revoked by several command-line options to another `netcon` session, or by `~@`, `~&`, `or ~*`, discussed in the list following this paragraph. Locked write permission is probably the mode you would want to be in if you're doing work on the console, as it can be revoked only by another session opening with a *-f* (force) option or with `~*`. The tilde (~) acts as an escape character for a `netcon` session. Table 17.1 lists several important commands that can be issued in this manner.

Table 17.1 netcon Commands

COMMAND	PURPOSE
`~#`	Analogous to Stop-A on a normal system. This will take the system down to OBP. Be careful when using this command, because its effects are immediate.
`~?`	Shows the current status of all the open `netcon` sessions by pseudo-terminal number.
`~=`	Switch between the SSP private interface for the domain and the control board JTAG interface. This feature works only in private mode.
`~*`	Private mode. This sets locked write permission, closes any open `netcon` sessions, and disallows access to `netcon` from any other terminal. This is the same as the -f option.
`~&`	Locked write mode. This is the same as opening a session with the -l flag.
`~@`	Unlocked write mode. This is eminently revocable by any other SSP user. This is the same as opening a session with the -g option.
`~^`	Read-only mode. Releases write permission (if currently set) and echoes any other session with write permission to your terminal.
`~.`	Release `netcon`. This will exit the netcon session and return you to the SSP command prompt.

What to Do If netcon Is Unavailable

There are situations in which the `netcon_server` may become unavailable or you find yourself unable to issue a `netcon`. If this is the case, first consult a `ps -ef | grep netcon_server` on your SSP. There should be one running for each domain that is up. This can be a little cryptic, because the only way to differentiate between domains is by viewing the -p *n* option on the `netcon_server` and deciphering which board that boot process is located on. Fortunately, we have included a handy formula for just this situation in the *Blacklisting for Fun and Profit* section later in this chapter. If you have determined that it is not running, then make sure your `$SUNW_HOSTNAME` is set to the domain you're attempting to connect to, and issue a `nohup netcon_server -r &` command. This option attempts to read the boot processor info from the SNMP daemon.

The Dynamic Domain

The more static features of the Starfire have been discussed thus far, but, by now, you're probably asking how all of this matters. Where, outside of the multiple domains, does the E10000 bring a strong advantage as opposed to multiple Enterprise systems? The answer lies within its dynamic reconfiguration functionality. There are two ways to access these features, either the pretty graphical (read: slow, buggy, and less than reliable) method, or the command-line interface. You've probably noticed our preference.

hostview—Since When Do UNIX People Need GUIs?

The GUI, named *hostview,* can perform many functions beyond dynamic reconfiguration that can be invoked from the command line, such as domain creation and removal, the opening of network consoles, and so on. The graphical interface also allows manipulation of the blacklist file for selected components, but we will delve into doing this manually later on. For simple components, such as an entire system board or a single CPU module, it may be easiest to use hostview to modify the blacklist. The reason for our negativity regarding this tool is that use of it has displayed numerous weaknesses. From our experience, it is notoriously flaky. It certainly has memory leaks, and will tend to eat free memory on your SSP if you leave it running. If you try to use it remotely (i.e., log in as ssp from your desktop workstation, export the display, and run hostview) it has a decent chance of crashing, and probably at a very inopportune time. Beyond that, it is slow and not terribly intuitive in some cases. Though most documentation you will find focuses directly on this tool, the scope of this section focuses on the command-line interfaces.

The dr Is In

The preferred method for reconfiguring your domains is the shell-like tool called `dr`. Like hostview and most other SSP binaries, it can only be invoked when you are logged in as the SSP user and will affect whatever domain is currently set in `$SUNW_HOSTNAME`.

To bring it up, simply invoke dr. You will be prompted with information that looks like this:

```
Ourobotos-ssp01:foonly-d1% dr
Checking environment...Establishing Control Board Server
connection...Initializing SSP SNMP MIB...Establishing communication with
DR daemon...
foonly-d1: System Status - Summary
BOARD #: 5 6 7 8 9 10 physically present.
BOARD #: 0 1 2 3 4  being used by domain foonly-d1.
dr>
```

You'll note that the interface presents itself as a shell. Commands passed in at this point affect only dr. At any time, you can enter *help* for a synopsis of commands. Here is a brief listing of the functions of dr, and the commands used to achieve them.

Attaching a system board:

init_attach *sysbd*. Begins the attach process for the specified system board.

complete_attach *sysbd*. Finalizes the board attach. This is only to be run after init_attach.**abort_attach *sysbd*.** Aborts the attach after init_attach, or a failed complete_attach.

Detaching a system board:

drain *sysbd*. Drains the memory on the system board to prepare it for detachment.

complete_detach *sysbd*. Finalizes a detachment. This is only to be run after drain.

abort_detach *sysbd*. Aborts a detach if drain or complete_detach fails.

TIP It is important to note that while dr will list all of the system boards not in the current domain as being physically present, this does not necessarily mean that they are eligible for attachment to the system. These may be used in running domains! Although the program should not allow you to actually attach a used board, it is suggested that you consult the output of domain_status before actually attempting an attachment.

The following listing shows a sample attachment. Note that a complete hpost is run on the board to be attached to ensure its hardware integrity. This is generally the only place that you should encounter a failure on an attachment; it is otherwise generally smooth.

```
dr> init_attach 5
Initiate board attachment, board 5 to domain foonly-d1.Adding board 5 to
domain_config file./opt/SUNWssp/bin/hpost -H20,0
Opening SNMP server library...
Significant contents of /export/home/ssp/.postrc:
#
```

```
logfile
Reading centerplane asics to obtain bus configuration...
Bus configuration determined to be 3F.
phase cplane_isolate: CP domain cluster mask clear...
<STANDARD HPOST OUTPUT (DELETED)>
phase final_config: Final configuration...Configuring in 3F, FOM =
2048.00: 2 procs, 4 Scards, 2048 MBytes.memboard_cage_swap(): INTERNAL:
memboard_sort_order not known.Creating OBP handoff structures...
Configured in 3F with 2 processors, 4 Scards, 2048 MBytes memory.
Interconnect       frequency is  83.284 MHz, from SNMP MIB.
Processor external frequency is 124.926 MHz, from SNMP MIB.
Processor internal frequency is 249.851 MHz, from proc clk_mode probe.
Boot processor is 5.0 = 20
POST (level=16, verbose=20, -H0020,0) execution time 3:08
hpost is complete.
/opt/SUNWssp/bin/obp_helper -H -m20
Board debut complete.
Reconfiguring domain mask registers.
Probing board resources.
Board attachment initiated successfully.
Ready to COMPLETE board attachment.
dr> complete_attach 5
Completing attach for board 5.
Board attachment completed successfully.
dr> exit
ouroboros-ssp01:foonly-d1% domain_status
DOMAIN          TYPE                     PLATFORM      OS         SYSBDS
foonly-d1      Ultra-Enterprise-10000   ouroboros      8      0 1 2 3 4 5
```

There is also a handy `dr` command called `reconfig`. Run from within the `dr` shell after a board attachment, it runs the standard Sun online configuration sequence on the domain—that is (in order), `drvconfig`, `devlinks`, `disks`, `ports`, and `tapes`. In an environment running Solaris 8 or later, it will invoke the `devfsadm` command. The same caution should be used as when running these commands manually, as they can move devices about.

Detachment is a bit more tricky, and the `drain` command deserves a special note. After running this command, `dr` returns immediately as if it has completed. *Do not* run a `complete_detach` immediately. `drain` can take from seconds to hours, depending on the usage of the system board you are trying to detach. The reason for this is that the command actually moves the pages of physical memory from the system board specified to any free memory it can find on the domain. If you're trying to remove the system board that has the kernel resident, it may seem as if it hangs forever. One instance had us waiting for three hours for the final 4K of memory to finish draining. The general rule for detaching a system board is that system usage must be generally low, unless the domain is exceptionally large. Do not expect to be able to reliably detach a system board from a two-board domain with a load average of 25. Also, keep in mind that the primary system board (the first one in the domain) can *never* be removed, as it contains the boot processor and kernel memory.

The command used to view the status of an in-progress drain is *drshow.* For the purposes of viewing a drain, the syntax is as follows:

```
drshow sysbd drain
```

The following listing shows a sample detachment.

```
dr> drain 5
Start draining board 5
Board drain started.
Retrieving System Info...
Bound Processes for Board 5
cpu   user  sys  procs
---   ----  ---  -----
20      0    2
22      0    2
No active IO devices.
                Memory Drain for Board 5 - IN PROGRESS
Reduction = 2048 MBytes
Remaining in System     = 12288 MBytes
Percent Complete        = 0% (2097152 KBytes remaining)
Drain operation started at Tue Aug 03 15:05:59 1999
Current time               Tue Aug 03 15:06:00 1999
Memory Drain is in progress...
When Drain has finished you may COMPLETE the board detach.
dr> drshow 5 drain
                Memory Drain for Board 5 - COMPLETE
Percent Complete        = 100% (0 KBytes remaining)
Drain operation started at Tue Aug 03 15:05:59 1999
Current time               Tue Aug 03 15:06:03 1999
dr> complete_detach 5
Completing detach of board 5.
Operating System has detached the board.
Reconfiguring domain mask registers.
Processors on board 5 reset.
Board 5 placed into loopback.
Removing board 5 from domain_config file.
Board detachment completed successfully.
dr>
```

Now for the caveats. While `dr` is an extremely powerful tool, great care should be taken in its use. In the early days of the E10K, we had crashed a number of production systems by using it wantonly or without proper research beforehand. You should *never* attempt to detach a system board with active I/O on it of any kind. It is fairly obvious that attempting to remove a board with an attached active disk array containing, say, your root file system would be an extremely bad thing. However, what are not so obvious are issues arising with I/O attached to a target system board even if it is *not apparently active.* Occasionally, domains decide to panic when `dr` attempts to detach something that they are not ready to let go of. For this reason, at our sites we instituted the convention of a *floater board* resource. What this means is that we have one (or two) system boards on the E10000 that are allocable. They have no I/O cards, just processors and memory. This way, we can bring the horsepower to whatever domain needs it, then detach it after the bulk of major processing is done with little (or no) problem. We would strongly suggest that if you are investigating the purchase of a Starfire, you equip it with similar boards. The headache reduction will be worth it.

There is a way from within dr to see if there is active I/O on a board. This is with another option to drshow, *io.* First, a board with I/O:

```
dr> drshow 0 io
                I/O Controllers and Devices for Board 0
------------------------- I/O Bus ) : Slot 1 : qec0 --------------------
---
device    opens  name                     usage
------    -----  ----                     -----
qe0              qe0
qe1              qe1
qe2              qe2
qe3              qe3
----------------------- I/O Bus 1 : Slot 1 : QLGC,isp0 -----------------
---
device    opens  name                     usage
------    -----  ----                     -----
sd0         25   /dev/dsk/c0t0d0s0        /
             0   /dev/dsk/c0t0d0s1        swap, /tmp
             8   /dev/dsk/c0t0d0s5        /var
sd1          0   /dev/dsk/c0t1d0s0
sd2          0   /dev/dsk/c0t2d0s0
sd3          0   /dev/dsk/c0t3d0s0
```

This board should *never* be detached. As for a candidate for detachment:

```
dr> drshow 4 io
                I/O Controllers and Devices for Board 4
No devices present.
```

this is a much healthier choice. Of course, I/O channels can be made quiescent on a system board if proper multipathing design was implemented when designing your domains. By utilizing features such as Veritas Dynamic MultiPathing, Solaris network multipathing, and similar utilities, one can create I/O channels that can be failed over to an alternate interface. If said interface resides on a different system board than the one that has been targeted for reconfiguration, the chances of successfully removing the board from an active domain increase significantly.

You should probably run the drshow command against any board you plan to detach, even if its I/O cards are completely disabled and/or physically disconnected from any cables. One may never know what the kernel could be holding open. If dr does decide that your reconfiguration choices are somehow inferior, and refuses to actually complete them, you can always bail out with the abort_attach or abort_detach commands. These are relatively nondestructive, and we have never run into a problem using them.

Don't let these warnings prevent you from using `dr`. Just make certain that your usage is logical, necessary, and does not interfere with the system's current functioning. If you're doubtful that there's enough memory available on the system to drain a board, or that the I/O is completely unused, don't attempt a detachment.

The practical upshot of this discussion is the elucidation of the advances made since the E10K server generation in terms of `dr` ability. Sun Fire servers, having finally divorced the I/O slots from the system boards, have few if any of the issues referred to in the above examples. `dr` becomes a far simpler issue in this newer generation of servers.

Blacklisting for Fun and Profit

Having perused dynamic reconfiguration, we should now touch upon not-so-dynamic configuration embodied in the blacklist. The blacklist is located in `$SSPVAR/etc /platform_name/blacklist`. This is read by `hpost` during a `bringup` or a `dr`. The syntax of this file is very straightforward, though the keywords may not be. Each line can have a case-insensitive keyword and a number or set of numbers. If this item is recognized, it will be ignored during `hpost` and left out of the physical configuration of the system once it is up. Here is a list of some important keywords:

sysbd *board_num.* Any physically present system board (0 to 15). This means that if the board is listed as being a member of a domain, it will be left out of the configuration. This will also prevent `dr` from adding the specified board to a domain.

mem *board_num.* The memory on the referenced system board.

scard *system_board.I/O_controller.slot.* The named sbus I/O adapter card. There are two I/O controllers on each system board, the upper half being 0, and the lower half being 1. The same is true for the slot, the upper one within the specified I/O controller being 0, and the lower being 1. So, if you wanted to ignore an Ethernet adapter in the third slot from the top of system board 3, the blacklist entry would be:

```
scard 3.1.0
```

proc *system_board.proc.* A single processor, listed by its number on the board. There can be up to four processors on each board, so the integer will be between 0 and 3. Example: *proc 5.2* would represent processor 2 on system board 5. Of course, since a panic string usually will list the CPU as an absolute number (i.e., CPU32), this notation is less than intuitive. In the past, this might have required the aid of a Sun service professional (since they possess the *secret* books and incantations), but instead here's a simple homebrewed formula to decode the `sysboard.proc` number from the absolute processor number, where *n* is the absolute `proc` number:

```
int( n / 4 ) = system_board_number the_integer_of_ndivided_by_four
n % 4 = processor_number nmodulofour
```

So, if the CPU listed in a panic string is CPU42, then the system board would be 10 (int (42 / 4)), and the processor number would be 2 (42 % 4). Your blacklist entry for this situation would be:

```
proc 10.2
```

Or, if you prefer hexadecimal:

```
proc A.2
```

There are also some more in-depth components that you can blacklist, but it is highly recommended that you know precisely what you're doing before attempting any of these:

COMPONENT	PURPOSE
mgroup ***board.bank*.**	The specified bank of memory (0 to 3) on a given system board.
mlimit ***board.group.mbytes*.**	Restricts the memory of a given bank to `mbytes`, either 64 or 256.
ioc ***board.io_controller*.**	An I/O controller on a given system board, 0 (upper) or 1 (lower).
xdb ***board.data_buffer*.**	The data buffer (0 to 3) on a given system board.
abus ***abus*.**	An address bus, 0 to 3.
dbus ***dbus*.**	Half of the centerplane data bus, 0 or 1.
ldpath ***board.dbus*.**	Half of the data bus on a given system board, 0 or 1.
pc ***board.controller*.**	A port controller, 0 to 2.
cic ***board.interface_controller*.**	A coherent interface controller (0 to 3) on a given system board.
cplane ***centerplane*.**	The half-centerplane, either 0 or 1.

The disadvantage of using the blacklist functions is that you will be unable to allow any of these components back into the system until another `bringup`. The one slight exception is that if you can safely `dr` the board in question out of the configuration (i.e., there is no attached I/O), remove the entries from the blacklist, then `dr` the board back in, you can bypass a `bringup`.

WARNING You will probably stumble across the `redlist` in your wanderings through the Starfire. This is a nearly useless file. It has the same syntax as the blacklist, and winds up behaving exactly the same, as well. It prevents the listed components from being tested by an `hpost`. Because only components that have passed `hpost` will be handed to the `obp_helper`, `redlisted` components wind up being ignored by the OS. This is probably a Sun engineering feature, and its use is strongly discouraged, because the way it is handled makes it far less stable than the blacklist. Your safest bet is to ignore it.

All of these features become especially useful when a hardware failure is encountered. An example scenario: A domain panics, and Sun determines that a CPU is the culprit and must be replaced. Instead of risking further downtime, you can blacklist the system board containing the CPU in question if the domain is down, or if you manage to get it back up, you can `dr` out the afflicted board if there is no dependent

I/O. This would then allow you to safely power down and remove the system board in question with *no further downtime.* In all, this is a very nice, very forgiving set of features, which is particularly pleasant when customers are constantly screaming about the <expletive deleted> downtime.

Recovering from Crashes

Now for the fun part—what does one do when all of this wonderful technology has utterly (and messily) failed? All machines are wont to do this at some point, so it is best to be armed with the proper information beforehand to ensure a quick and smooth recovery.

Before diving into situations, here is a look at the tools you can use. These should obviously be used with caution; they can and will perform their functions on a running domain as invoked. These commands all affect the domain set as `$SUNW_HOSTNAME`. They are listed in the order in which they should be attempted, from least severe to most:

hostint. Processor interrupt. This will force a panic on the domain, which will dump before rebooting.

hostreset. This sends an externally initiated reset to the domain, an action much akin to switching the power key off on an Enterprise system. The domain then goes into limbo, but you should then be able to run a `bringup` on it. This is also useful at the OBP level—if you are hung waiting for a boot, it can be used to get you back to the ok prompt.

sys_reset. This command is very, very, harsh. This resets all of the system boards in the current domain. This will not work against a running domain without the -f (force) option. The system will be left in limbo, and you will probably need to run a `bringup`.

bringup. Our old friend, with a new option, -f (force). This will run the `bringup` even if the host is up. Do not initiate with this option in any but the most extreme situations.

The most common problem for which these commands may be useful is a hung domain. Normally, if an event is discovered that should cause a panic, the EDD will initiate a reboot. If for whatever reason this does not happen, your domain will hang. When this occurs, the goal is to recover with minimal system impact and data loss. Use of these commands assumes that you are unable to reboot the machine by normal means, and that issuing a Stop-A (`~#` from `netcon`) will not work.

If your domain panics, there is a bit more to do than if a normal Enterprise system were to panic. We would tend to insist that every domain on your E10000 have `savecore` enabled, because it is direly important to have panic dumps from these machines. Now, the first step in recovery is readily apparent: Get the system back up. Hopefully, this should be accomplished with the commands already given. If the crash has rendered your system unusable, general Solaris procedures are to be followed for recovery of the actual domain. In any case, Sun will also need any messages files from `$SSPVAR/adm` and `$SSPVAR/adm/domainname`. The most important component, however, is the collection of dumps located in `$SSPVAR/adm/domainname`. Here are the various types you may encounter, all of which will be time and date stamped:

Edd-Arbstop-Dump. This is a fatal error, usually caused by a CPU panic. When this is created, EDD will try to do a complete `bringup` on the domain after the dump completes.

Edd-Record-Stop-Dump. These errors are actually fairly common. They are often left by a crash, but they can be caused by ECC memory errors and other correctable conditions, as well.

hostresetdump. This is the dump left behind by the execution of a hostreset on a domain.

watchdog-redmode-dump. The dump when a processor enters a watchdog condition. This will cause EDD to do a complete `bringup` on the domain.

When a crash is encountered, any of these files, along with the message files (both SSP and domain-specific), POST output, `bringup` output, and anything else you have become relevant. One thing we always try to do is capture the console output, so we generally leave one `netcon` session open per domain on the SSP. Even if the write access to that `netcon` is stolen away when we access it remotely, we still have a record that we can scroll back to and copy to a text file when next we visit the SSP.

A very important hint to keep in mind is that the SSP is the hub of all Starfire activity, and can in fact get itself into strange states. The thing is, you can reboot it at just about *any time.* The only ill effect you will notice is that the sudden loss of fan control will cause a sound like a jet engine to emanate from your E10000. It is certainly not recommended that you leave it down for extended periods of time, but we cannot tell you how many times rebooting the SSP has actually cleared a condition that had every sysadmin in the vincinity dumbfounded. Keep in mind, there will be no error detection or environmental control while the SSP is rebooting.

We have also encountered more esoteric problems on the Starfire. Once, a domain crashed, and every time we tried to run a `bringup`, the `bringup` itself would crash and attempt to `redlist` components. Odd behavior, indeed. Eventually, we deciphered the fact that it was attempting to `redlist` a particular system board's ASICs, powered off that offending board, and powered it back on. This did indeed work. Often, if a problem like this is encountered, it is best to analyze any data and draw the best conclusion you can from it. Then, step back and assess what (if any) action can be taken. Keep all of the various aspects of this often complex machine in mind, and you will find yourself coming up with innovative solutions. Of course, there are the occasions on which hardware does fail catastrophically, and in these cases it is best to rely on your blacklisting or `dr` skills. Avoiding downtime is the key.

Above all else, we recommend getting any information available (cores, logs, etc.) to Sun as soon as possible after the dump. They tend to be extremely responsive and helpful on any problem that you can encounter on a Starfire.

Summary

The Enterprise 10000 is a very unique and interesting system, with a number of eccentricities. This alternative to a mainframe allows you to:

- Add processors
- Add memory and I/O
- Take down portions of the machine without bringing down the whole thing

If you are fortunate enough to have the E10000, don't let its radical design or the differences between how this machine and other Sun systems are managed keep you from appreciating what this system can do for you. In the most data-intensive environments, the Starfire's speed and reliability will *make* you love it.

CHAPTER

18

The Sun Fire Servers

Building on the success of the Starfire (see Chapter 17), the line of servers that introduced dynamic reconfiguration and capacity on demand, Sun now offers a new line of servers—the Sun Fire servers. When the first edition of this book came out, the Enterprise 10000 (Starfire) was the most amazing system we had ever seen. Well, we have been amazed once again. This new family of servers offers many technological advances over the UltraSPARC II family of servers. In this chapter, we take a look at the architecture of this new product line and explain how the new features provide an unprecedented level of reliability, availability, and serviceability (or *RAS*). We explain how RAS is incorporated into the basic design aspects of this product line to contribute to a higher level of operation and a reduction in downtime for the services that they provide.

We will also offer suggestions on how to administer and troubleshoot basic problems on these systems.

Introduction to the Sun Fire

The Sun Fire Server family is well suited to provide high-performance computing for many types of businesses, including:

- Internet service providers (ISPs).
- Electronic Commerce (eCommerce).

- Corporate and Technical Data centers.
- Financial services, including banking, brokerage, insurance customers, billing services, and claims processing.

In fact, there are literally thousands of business applications that will run on the Solaris Operating Environment making these systems ideal for many business requirements. The following sections discuss the hardware behind Sun's latest line of servers.

A New Architecture

The Sun Fire servers incorporate a new hardware architecture that offers superior performance using the new third generation 64-bit UltraSPARC III and UltraSPARC III CU processor design. This processor has a higher clock speed than previous UltraSPARC processors, starting at 750 MHz and projected to go as fast as 1.8 GHz. The processor design also allows for greater scalability—enabling hundreds of processors to be used simultaneously.

The new Fireplane interconnect technology greatly increases total bandwidth in the server. Performance over the Gigaplane architecture in the Enterprise Server line and the E10K server is dramatic. The switch technology reduces data bandwidth bottlenecks and allows the various busses to operate at near 100 percent capacity at all times.

Data reliability is enhanced in this design because many of the new application specific integrated circuits (ASICs) are protected by error correction code (ECC) and/or parity-checking technologies. ECC is used on the external processor interfaces in order to detect and correct single bit errors. Parity error detection is used on every external chip-to-chip hop that the data travels through in order to isolate faults. Because the hardware is designed to test data as it is traveling from point to point, the system can be assured of reliable data.

Many of the critical components are redundantly built into the system. Engineered into the firmware is an improved power-on self test (POST; see the *POST* section later in this chapter). Under the default configuration, all critical components are tested to ensure that they are functioning properly when the server is powered on or rebooted. Enhanced error messages reduce the guesswork as to what is causing a problem. Many of the ASICs also have a technology called *built-in self test* (BIST), which provides continuous testing on the ASIC while the system is functioning. In addition, this modular design, utilizing fewer *field replaceable units* (FRUs), makes field service easier for qualified technicians. On some systems, components are hot-pluggable or hot-swappable. This allows you to add, remove, or replace the component while the system is running.

The Sun Fire architecture also provides backward compatibility with existing hardware and software, extending the usefulness of those products. All SPARC processors are designed to ensure binary compatibility across generations and platforms. This architecture is based on the SPARC IEEE 1754 open standard specification. The 64-bit design incorporated in the UltraSPARC III CPUs is an extension of that specification.

For more info on the SPARC specification see www.sparc.org.

Sun Fire Technical Specifications

Let’s take a look at the key technologies built into the hardware on these servers. The Sun Fire Server family is composed of these server lines and models:

- The Sun Fire Workgroup Servers (the 280R, V480 and V880)
- The Sun Fire Midframe (midrange) Servers (the 3800, 4800, 4810 and 6800)
- The Sun Fire High-End Servers (the 12K and 15K)

At the low end are the Sun Fire workgroup servers. These systems are marketed as the low-priced *symmetric multiprocessing* (SMP) workgroup server systems. SMP systems are designed to spread the system load across many processors. Multithreaded applications in particular take advantage of this feature by allowing each thread to operate at the same time over the multiple processors. This design is a major improvement over asymmetrical processing, because any task can run on any idle processor, and additional CPUs can be added to improve performance and handle increased processing loads. Later in this chapter we will show you how the Fireplane interconnect provides high bandwidth to memory for increased performance. SMP also allows a single operating environment (OE) to share disk I/O resources and up to 8 GB per processor of common memory.

At the middle of the product line are the Sun Fire midrange servers. These systems are marketed to provide capabilities previously found only in mainframe servers. They also utilize symmetric multiprocessing (SMP) technologies. Midrange systems are designed to provide higher availability than workgroups servers and offer remote administration features. They also support multiple independent operating environments running simultaneously (these are known as *domains*).

At the high end of the product line are the Sun Fire high-end servers. These two servers extend the feature sets of the midrange servers and provide much more scalability (up to 106 processors in a single server). Tables 18.1, 18.2, and 18.3 compare the different Sun Fire servers.

The Sun Fire line of servers provides many of the same features (such as dynamic reconfiguration), while covering a much wider performance range.

Table 18.1 A Technical Comparison of Sun Fire Workgroup Servers

HARDWARE COMPONENT	SUN FIRE 280R	SUN FIRE V480	SUN FIRE V880
CPUs and speed	1-2 @ 750 or 900 MHz	2 or 4 @ 900 MHz	2, 4, 6, or 8 @ 750 or 900 MHz
Memory (Max GB)	8	32	64
E-cache (MB) per processor	8	8	8
Max PCI I/O slots	4	6	9

(continues)

Table 18.1 A Technical Comparison of Sun Fire Workgroup Servers *(Continued)*

HARDWARE COMPONENT	SUN FIRE 280R	SUN FIRE V480	SUN FIRE V880
Max cPCI I/O slots	N/A	N/A	N/A
Dynamic system domains	1	1	1
Internal mass storage	(2) 36 GB or 73 GB, 10,000 RPM FC-AL disk drive, CD-ROM/ DVD-ROM	(2) 36 GB or 73 GB, 10,000 RPM FC-AL disk drive, DVD-ROM	(12) 36 GB or 73 GB, 10,000 RPM FC-AL disk drive, CD-ROM/ DVD-ROM
Redundant components	Hot-pluggable power supplies	Hot-pluggable power supplies	Hot-pluggable power and fan supplies
Supports dynamic reconfiguration	No	No	Yes

Table 18.2 A Technical Comparison of Sun Fire Midframe Servers

HARDWARE COMPONENT	SUN FIRE 3800	SUN FIRE 4800/4810	SUN FIRE 6800
CPUs and speed	2-8 @ 750, 900 or 1,050 MHz	2-12 @ 750, 900, or 1,050 MHz	2-24 @ 750, 900, or 1,050 MHz
Memory (Max GB)	64	96	192
E-cache (MB) per processor	8	8	8
Max PCI I/O slots	N/A	16	32
Max cPCI I/O slots	12	8	16
Dynamic system domains	1-2	1-2	1-4
Internal mass storage	None	None	None
Redundant components	Hot-pluggable power supplies	Hot-pluggable fan supplies	Hot-swappable system boards
Hot-swappable I/O boards	Hot-pluggable power and fan supplies, system and I/O boards	Hot-pluggable power and fan supplies, system and I/O boards	Hot-pluggable power and fan supplies, system and I/O boards
Supports dynamic reconfiguration	Yes	Yes	Yes

Table 18.3 A Technical Comparison of Sun Fire High-End Servers

SUN FIRE HIGH-END SERVERS	SUN FIRE 12K	SUN FIRE 15K
CPUs and speed	2-52 @ 900 or 1,050 MHz	2-106 @ 900 or 1,050 MHz
Memory (Max GB)	288	576
E-cache (MB) per processor	8	8
Max PCI I/O slots	32	64
Max cPCI I/O slots	N/A	N/A
Dynamic system domains	1-9	1-18
Internal mass storage	None	None
Redundant components	Hot-pluggable power and fan supplies, System, I/O and expansion boards	Hot-pluggable power and fan supplies, System, I/O and expansion boards
Supports dynamic reconfiguration	Yes	Yes

The Sun Fireplane Interconnect

As we explained in Chapter 10, the key to maximum performance is to eliminate bottlenecks in the operation of the hardware. In the PC world, vendors have focused on CPU clock speed because, up until recently, PC CPUs could only run one transaction at a time. To understand what this means, think of a transaction as being one step (or instruction) in a math problem. Therefore, if a system can provide more cycles per second, it can process more transactions per second. Sun Microsystems' RISC-based SPARC design does just this. It allows for multiple transactions per second for each processor, and SMP allows for transactions from the same process to be handled by multiple processors at the same time.

So if you make a faster processor capable of processing multiple transactions at the same time, where is the bottleneck? The bottleneck exists when you try to access memory. Over the years, the number of instructions processed per second has increased dramatically. This is measured as millions of instructions per second *(MIPS)* or recently as billions of instructions per second *(BIPS)*. The ability to save and retrieve this information in memory, on the other hand, has not increased at the same pace. This delay in the time that it takes to request a memory location and for memory to respond to a request for the data stored at that specific location is known as *latency*. This is illustrated in the following Figure 18.1.

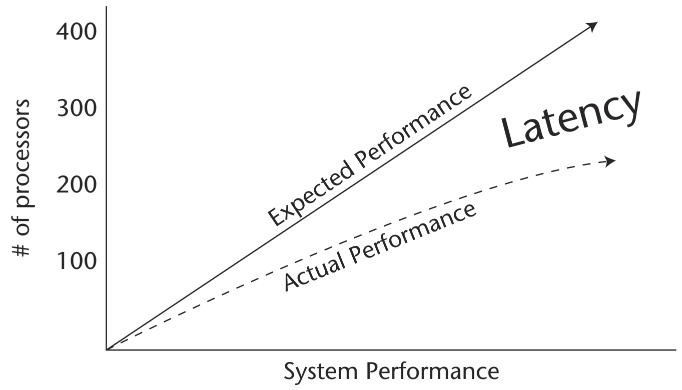


Figure 18.1 Latency.

Now, let's examine what this means. You would expect to see twice the performance using 100 processors as you would using 50, or eight times the performance when using 400 processors. But memory can't keep up with the number of requests these processors can generate. This bottleneck is addressed in the design of the UltraSPARC III CPU and the Sun Fireplane Interconnect.

The Sun Fireplane Interconnect is composed of a number of internal busses and chips that control the mechanism for accessing and transferring data from one resource to another. Because the key to higher performance is to reduce bottlenecks, the Sun engineers designed this interconnect to reduce latency by separating the address and control functionality usually built into the same circuitry into their own dedicated buses. The advantage of this design is that it eliminates a centralized bus arbitrator because the arbitration for the address and control lines is performed simultaneously by all devices. In other words, when a processor requests a virtual memory location, the request is heard by all of the *memory management units (MMU)* located in each CPU, and the request to transmit the actual data from its physical location is then processed. As a result of isolating the circuitry to request data and sending information regarding the location of that data from the circuitry that the data actually travels on, performance increases greatly.

The interconnect design is basically the same throughout the Sun Fire product line, with the larger systems having more levels of switching to enable more data to be processed and isolation of resources when configured in different domains. The data path is 288-bits wide between the UltraSPARC III CPUs and the *I/O controller chip (IOC)*. This architecture is designed for a point-to-point connection between these components and the memory subsystem.

The UltraSPARC III data port is a private interface that requires a data switch. On the Sun Fire 280R, each CPU card is interfaced to a *combined processor-memory switch (CPMS)* port through a private data bus. This data bus interfaces to the CPMS chips that are resident on the motherboard. Each CPU private data bus is made up of a 144-data-bit-wide bus that has 16 bits for ECC and 128 bits for data. This bus is parity protected.

On the Sun Fire V480 and V880, each CPU/memory board contains two UltraSPARC III CPUs. The CPUs are connected to a data switch known as the *dual CPU data switch* (DCDS). There are eight of these application specific integrated circuits (ASICs) on the CPU/memory board. They provide Level 0 switching between the CPUs and their dedicated memory banks. The DCDS has five ports, one for each of the CPUs it services, one for each of the dual memory banks dedicated to each of these processors, and one for the *multiport data repeater/8-way (MDR8)* ASIC located on the motherboard. The port to the DCDS can process up to 4.8 GB per second, and each of the other ports can processes up to 2.4 GB of data per second going to the IOCs. Also located on the motherboard are two ASICs that provide address and control capabilities. The *data address repeater/8-way (DAR8)* ASIC provides address and arbitration services and the *data control switch/8-way (DCS)* ASIC provide control capabilities from the CPU to the IOCs.

On the Sun Fire midrange and high-end servers, the CPUs and memory are located on a hot-swappable board known as a *system board*. Either two or four CPUs are installed at the factory on these boards, and they support up to 32 GB of memory per board. The only FRU on the board is the memory. CPUs are not replaced in the field anymore. These boards also use the DCDS ASICs to provide the data port to other resources in the server. The DCDS provides each pair of processors with access to the Fireplane interconnect. This is known as *Level 0 switching* on the Sun Fireplane Interconnect.

The system boards also use two additional ASICs to allow each CPU/memory pair on the board to communicate with each other as well as with additional system boards in the server. The *address repeater* (AR) ASIC and the *data switch* (DX) ASIC. Because the DX ASIC is connected to each of the DCDS ASICs, as much as 9.6 GB per second of data can be transferred between processors and their dedicated memory subsystems on a system board. These two ASICs provide Level 1 switching on the Sun Fireplane Interconnect.

The AR ASIC is responsible for address bus communications. This allows CPUs within a board to communicate with each and make requests for data, and it allows other system boards address buses to make requests to each other.

Each system board also contains a *Sun Fire data controller* (SDC) ASIC, which allows the *bootbus* (an 8-bit bus used when POSTing the domain) and the *console bus* (used for administration) to communicate with the system controller. We will investigate this in the *POST* section later in this chapter.

On the midrange servers, Level 2 switching on the Sun Fireplane Interconnect is provided by the *switch boards* (known as RP0, RP1, RP2, and RP3 on the Sun Fire 6800; RP0 and RP2 on the Sun Fire 3800, Sun Fire 4800, and Sun Fire 4810).

Each switch board has the DX, AR, and SDC ASICs. These three levels of switching provide a way for each resource (CPU, memory, and IOC) to communicate with every other resource. It also provides a means to isolate resources from each other.

The Sun Fire 12K and 15K use the same system boards as the midrange servers. Level 0 and Level 1 switching on the Sun Fireplane Interconnect works the same way on these systems. Level 2 switching is provided by a different board known as an *expander board*.

On these two systems the system boards are installed in slot 0 of an expander board. The I/O boards are installed in slot 1 on the expander board. The expander board acts as a 2 to 1 multiplexor. In other words, it allows two busses to be combined into one bus. It has a *system address controller* (SAC) ASIC used to provide address and control services and a three port *system data interface* (SDI) ASIC for access to the data bus. These two ASICs provide Level 2 switching on the Sun Fireplane Interconnect so that resources in slot 0 (the system board) and resources in slot 1 (the I/O board) can communicate with each other. It also provides a communication path to Level 3 switching on the Sun Fireplane Interconnect. The expander board and the slot 0 and 1 boards installed in it are referred to as a board set. The maximum transfer rate from one board set to the other board sets is 4.8 GB per second.

Level 3 switching is located on the centerplane. Nine board sets connect to each side of the Sun Fireplane along with one system controller board set. The Sun Fire 15K has a 18 by 18 crossbar switch that provides up to 43 GB per second of data through the centerplane. The Sun Fire 12K has a 9 by 9 crossbar switch, providing half as much throughput. The address bus utilizes an ASIC called the *address extender queue* (AXQ), and the data bus uses a *data multiplexor* (DMX) ASIC. The Sun Fireplane crossbar switch allows for the creation of up to 18 separate domains, each running an independent copy of Solaris, or any combination using the available resources.

CPUs and Memory

The UltraSPARC III and UltraSPARC III CU CPUs use 64-bit virtual addressing and have very high integer and floating point performance. They also have a larger cache and lower cache latency than the UltraSPARC II processor. This is a major improvement.

On all of the Sun Fire servers, the UltraSPARC III chips have cache buses that are composed of internal caches (icache and dcache) that are located in each of the UltraSPARC III chips. The internal cache (L1 cache) is made up of 32 KB of *instruction cache* (icache) and 64 KB of *data cache* (dcache). Locating the cache on the CPU itself speeds up performance. In addition to internal cache, there is *external cache* (e-cache—also known as L2 cache) that consists of 8 MB of high-speed memory. The e-cache uses a 288-bit-wide data bus that has 32 bits for ECC and 256 bits for data.

When looking for ways to improve performance, the engineers at Sun needed to address the bandwidth of the data bus. In most computer architectures, what limits the data bandwidth isn't the speed of the bus, but rather the number of requests for data that can be processed. The number of requests generated is limited by how many devices can be connected to the bus.

Consider what happens when a CPU takes data from memory, copies it to its internal cache, and then changes that data as needed by the process it is supporting. The current copy of the data now resides in internal cache not in physical memory. What happens when another CPU or an IOC needs to access that same data? It would need to access the current copy of the data from the CPU's internal cache. Maintaining cache coherency is an issue when dealing with this scenario.

The Sun Fireplane address bus circuitry solves this problem by broadcasting a request for data by a CPU or IOC, and every device on the address bus reviews that request to determine if it is the device that controls access to the data in its cache. If it is the device, then it sends the data to the requesting device. This is known as *bus snooping*, which uses a series of protocols to allow the latest copy of the data to be available at all times, and the technology known as *cache coherency* makes sure that the old copies of the data are discarded.

On the Sun Fire 12K and 15K, Level 0, 1, and 2 switching on the Sun Fireplane Interconnect use the snoopy coherency technologies. At Level 3 on the centerplane, a different communication protocol known as *scalable shared memory* (SSM) is used between the different board sets. SSM does not have the overhead that bus snooping has.

Consider a domain that consists of one expander board, one slot 0 system board, and one slot 1 I/O board. All requests for data and the transmission of that data will reside within the board set. In other words, they never need to travel across the centerplane.

On each expander board is a special cache called the *coherency directory cache* (CDC). If the domain uses multiple board sets, the off-board request is checked with the CDC to determine if the CDC knows the location of the data. If the CDC knows the location, it reports that information back to the requesting device. If the CDC doesn't know where the data is located then, based on the request address, the board containing the data is passed the request.

These technologies allow for almost 100 percent full utilization of the data bus bandwidth throughout the server.

The Sun Fire 280R uses a CPU card for each of the two possible CPUs that can be installed on the system. The memory is located in DIMM slots on the motherboard.

The Sun Fire V480 and Sun Fire V880 use a CPU/memory board that holds two CPUs and up to 16 GB of memory per board. The Sun Fire V480 supports up to two CPU/memory boards, and the Sun Fire V880 supports up to four of them.

The Sun Fire midrange servers use system boards that support either two or four CPUs, and each board supports up to 32 GB of memory that can be interleaved for faster access. The Sun Fire 3800 supports up to two system boards, the Sun Fire 4800 and Sun Fire 4810 up to three system boards, and the Sun Fire 6800 up to six system boards.

The Sun Fire high-end servers use the same system boards as the midrange models. The Sun Fire 12K supports up to nine system boards, and the Sun Fire 15K up to eighteen. They also support a special slot 1 board called the *MaxCPU Board*. These boards fit in slot 1 and contain 2 CPUs but no memory. This allows the Sun Fire 15K to scale up to 106 processors (18 system boards with 4 CPUs per board, plus 17 MaxCPU boards with 2 CPUs per board). The Sun Fire 12K can use up to 8 MaxCPU boards.

I/O

The IOC ASIC controls access to PCI I/O cards, or, in the case of the midrange servers, hot-swappable *compact PCI* (or cPCI) I/O cards. SBUS cards are not supported on this family of servers because the sustained data rate on the SBUS technology is only 65 MB per second, which is too slow for today's needs.

The IOC ASIC has a maximum data transfer of 1.2 GB per second to the DX ASIC, and it has two buses that connect to the card slots. Bus A is a 64-bit, 66/33 MHz bus and supports high-speed *Enhanced Peripheral Component Interface* (EPCI) card slots. A Bus A slot can transfer 400 MB of data to the IOC per second. The Bus B is a 64-bit, 33 MHz bus and supports slower-speed PCI card slots, each capable of moving 200 MB of data to the IOC per second.

The Sun Fire workgroup servers have the IOC ASICs located on the motherboard. The Sun Fire 280R uses one IOC to support one EPCI and three PCI card slots. The Sun

Fire V480 uses two IOCs to support two EPCI and four PCI card slots. The Sun Fire V880 uses two IOCs to support two EPCI and seven PCI card slots.

The Sun Fire midrange servers have the IOC ASICs located on I/O boards. They are I/O board slots on the servers referenced as IB6, IB7, IB8, and IB9 on the Sun Fire 6800 and as IB6 and IB8 on the Sun Fire 3800, Sun Fire 4800, and Sun Fire 4810.

The Sun Fire 3800 only supports cPCI I/O boards. These boards contain two IOCs that support two EPCI cPCI card slots and four PCI cPCI card slots.

The other three models support both PCI and cPCI I/O boards. Because the I/O board slot in each of these three systems has a different-sized footprint, the cPCI I/O board can only support four cPCI card slots. Two IOCs support two EPCI and two PCI card slots on the cPCI I/O board. The PCI I/O board for these three systems contains eight slots supported by two IOCs. Two of the cards slots are EPCI, and the other six are PCI.

Because all I/O boards on the Sun Fire midrange servers have two IOC ASICs, each capable of a maximum of 1.2 GB per second data bandwidth, the total data accessible in a second is 2.4 GB.

The Sun Fire high-end servers support three different slot 1 boards, described in the following list. The first two provide I/O capabilities.

- hsPCI I/O Board
- Sun Fire Link Boards
- MaxCPU Board (CPU's only-no I/O)

The *hot-swappable PCI* (hsPCI) I/O board has two IOCs and supports four hot-swappable cassettes, two for EPCI and two for PCI cards. The hot-swappable cassettes allow you to remove and reinsert the cassettes while the operating environment is running. The Sun Fire 12K supports up to nine slot 1 hsPCI boards, and the Sun Fire 15K supports up to 18 slot 1 hsPCI boards.

A current listing of all supported I/O cards is located on the online Sun System Handbook site at http://sunsolve.sun.com/handbook_pub/.

Another slot 1 I/O board is the Sun Fire link board. This new product provides a fiber-based intelligent interconnect that allows multiple Sun Fire midrange or high-end servers to be connected together. The board looks like the hsPCI I/O board, except that two of the cassettes contain fiber-optics cards that provide a point-to-point connection with up to three other servers.

These cards are connected to a special 12-strand fiber cable that connects to the Sun Fire link switch. Each cable can be up to 75 meters long and contains multimode fibers that operate at 1.2 GB/sec. Ten of these fibers are used for data, one as a 600-MHz clock, and the last one for framing to keep all the signals in sync. The total bandwidth of the board is 2.4 GB/sec because of the slot 1 board Sun Fireplane Level 1 bandwidth limitation. The other two slots are regular hsPCI card slots. There is a version of the Sun Fire link board for the midrange servers as well.

The Sun Fire link board can be used to grow your domain past the limitations of the server. For instance, if you had a 24-CPU Sun Fire 6800 and needed more processing power, you could use the Sun Fire link board to connect another Sun Fire 6800 to the one you already have, and now your domain could scale to 48 CPUs.

You could also use it for clustering using Sun's Cluster 3.0 software. This would provide you with much faster response time if a node failed in the cluster.

Integrated Service Processors

The midrange and high-end Sun Fire servers support two integrated service processors. They are known as *system controller* (SC) boards. They contain the hardware to provide the following:

- A programmable system and processor clock to synchronize all of the buses throughout the server.
- A network interface for remote administration connectivity.
- Configuration and administration of the server's environmental settings.
- Configuration and administration of the domain settings.
- Coordination of the boot process.
- Environmental sensor monitoring.The sensors send signals back to the SCs on the I^2C bus. The I^2C bus is a simple three- wire 8-bit bus used to communicate the current temperatures, voltages, amperages, fan speeds.
- Control and monitoring of the power supplies.
- Error analysis and Automatic System Reconfiguration (ASR). If a component fails during POST it is marked out of use and the appropriate actions are taken to reconfigure the domain (if possible).
- Console functionality for remote administration.
- Automatic failover and failback capabilities between system controller boards-if one SC fails, the other takes over immediately. Running domains are unaffected.
- Centralized time of day. Each domain initially gets its current time and timezone from the SC.
- Centralized reset logic; if a SC fails it can be rebooted.

There are up to two SCs on the midrange Sun Fire servers, and they do not have disk drives for storage, therefore they cannot store Solaris them. Instead they run a *Real Time Operating System* (RTOS) called *SCApp* that provides a minimal Unix-like environment. SCApp is stored as firmware on the SC and provides the code necessary to boot and test the SC, analyze the current configuration of the server, and configure the resources (system boards and I/O boards) in their designated domains. They are designated on the servers as SC0 and SC1, and contain an Ethernet network connection as well as a serial port connection (to provide console access). On the SC board itself are additional ASICs that provide environmental sensing and monitoring, error detection and error logging, bootbus support (we talk about this in the *POST* section later in this chapter), firmware, interboard communications, heartbeat signal to the other SC, clock frequency generation, and various low-level functions.

There are up to two SCs on the high-end Sun Fire servers, and they do have disk drives for storage, therefore they can store and run either Solaris or Trusted Solaris on them. They provide the same type of services that the SCs do on the midrange Sun Fire servers and offer a higher level of authentication and security because they are running a full version of Solaris. The SC is composed of a Sun Microsystems CP1500 computer with a 440-MHz UltraSPARC Iii processor (equivalent to an Ultra 10 system) that is mounted on a cPCI hot-swappable card slot. The CP1500 is mounted with additional circuitry on a control board that is the same size as a slot 0 board. The CP1500 supports up to 1 GB of memory. One of the SCs is located on the front far-right expander board slot, and the other one is in the same place on the rear side. They are designated on the servers as SC0 and SC1 and contain two Ethernet network connections (for redundancy) as well as a serial port connection (to provide console access). The SC board itself consists of additional ASICs that provide the same type of services as those listed in the previous paragraph.

The SCs on all of these servers provide a console connection to all of the domains as well as a communication path to all of the hardware in the server. This allows you to gather the status of a specific board or component even if it is not in use or if the primary data bus has failed in a domain. It also allows you to test various components without having a running domain or to test them while the domain *is* running using the dynamic reconfiguration technology. On the high-end Sun Fire servers, this concept goes a step further through the provision a private network for all the domains that is maintained by the SCs. These private networks are called *management networks* (MANs). (There are no MANs on the low-end or midrange servers). See the *Management Networks* and *System Management Services* sections later in this chapter for more information on SMS and MANs.

Operating Environment Requirements

The Sun Fire family of servers requires as a minimum Solaris 8 and are also compatible with Solaris 9. To determine which release version is installed on a Sun server run this command:

```
# more /etc/release
```

Table 18.4 lists the minimum release version supported for each server.

Table 18.4 Solaris Release Requirements

SUN FIRE SERVER	MINIMUM SUPPORTED RELEASE
Sun Fire 280R	Solaris 8—1/01 and patches (see http://sunsolve.sun.com/data/816/816-1712/pdf/)
	Solaris 8—7/01
	Solaris 8—10/01 is minimum for 900 MHz
	Solaris 9
Sun Fire V480	Solaris 8—2/02 and patches (see http://sunsolve.sun.com/data/816/816-0905/pdf/)
	Solaris 9
Sun Fire V880	Solaris 8—7/01 for 750 MHz CPUs
	Solaris 8—10/01 and patches for 900 MHz CPUs
	Solaris 8—2/02 for 900 MHz CPUs
	Solaris 9

Table 18.4 *(Continued)*

SUN FIRE SERVER	MINIMUM SUPPORTED RELEASE
Sun Fire3800, 4800, 4810, and 6800	Solaris 8 — 4/01- is minimum OS for 750MHz modules
	Solaris 8 — 10/01 for 900 MHz and 1050 MHz
	Solaris 8 — 2/02 for Dynamic Reconfiguration
	Solaris 9
Sun Fire 12K and 15K	Solaris 8 — 10/01
	Solaris 9

The Sun Fire Boot Process

Solaris starts up the same way on the Sun Fire server family as it does on any other Sun system. It is important to remember that the midrange and high-end servers must be connected to a bootable device because they do not contain internal boot drives. The OBP `auto-boot` variable needs to be set to `true` in order for the server (or domain) to boot automatically. The installation process is also the same on these systems. Sun recommends using a Jumpstart server for the midrange and high-end servers to simplify installation. Sun's new Flash technology can speed up this process as well. Refer to Chapters 4 and 5 for more information on the boot processes.

Although you may be familiar with the POST and PROM settings for older systems, the Sun Fire line of servers expands on these features with new options.

POST

The power-on self test (POST) is designed to identify all of the components up to the card slots in Sun Fire servers. POST will also test the various ASICs to determine if they are functioning properly. The output of POST can be seen using a serial connection to the console port when the system (or domain) is powered on or reset.

The sequencing of POST is system dependant. In other words, the Sun Fire 280R's POST uses different tests in a different order than the ones used on a Sun Fire 6800.

The output of POST can be cryptic and is used primarily by Sun support. Documentation is not available to the public as to what is done with each test. And why would we need to know exactly what each test does? Instead, we should be concerned that each test *passes*. If a component fails, the hardware will mark it out of use and reconfigure the system. If it is a critical component on a board, then the board might be marked out of use. If it is a vital system component that would prevent the system from booting, then POST will fail. The failure messages are what we are looking for in the POST output. The `syslog` logging function will log these messages in `/var/adm/messages` by default on the workgroup servers.

Remember that the midrange servers' SCs can only store about two days worth of logging messages for the SC and each domain. So, it is very important to configure `syslog` message forwarding using the `loghost` setting with the `setupplatform` command in the platform shell and the `setupdomain` command in each domain shell. You set the IP address of the system you want to forward the `syslog` messages to and the facility you want to use (the facility represents a category to allow grouping of messages for syslogd—see the man page for `syslog` for more info on this). Sun recommends using the `local0` facility for the SC, `local1` for domain A, `local2` for domain B, and so on.

The high-end servers have a background process called the Message Logging daemon (`mld`) that is responsible for logging POST error messages.

Each of the servers allow for a higher level of testing to be done when you suspect you have a hardware problem. POST assures us of a clean snapshot of the system configuration when bringing up the OE.

Open Boot PROM

The Open Boot PROM concept is based on another industry standard developed by Sun. It is the "IEEE 1275-1994 Standard for Boot (Initialization Configuration) and Firmware Core Requirements and Practices."

This standard describes a software architecture for the firmware that controls a computer before the operating system has begun execution. It is critical that the hardware configuration be defined so that the software drivers can correctly operate and communicate with the various hardware components.

The firmware located in the flashprom(s) on the server contain the code to identify the devices, initialize them at a low-level hardware state, gather information on their current configuration, and test them.

This standard was designed to not be vendor specific or architecture specific. It is a shame that the rest of the industry does not use it! The Sun Fire workgroup and high-end servers use the OBP 4.x firmware, and the Sun Fire mid-range servers use OBP 5.x firmware.

OBP 4.x and 5.x mostly use the same OBP commands as the 3.x version. The difference between OBP 3.x and OBP 4.x firmware is that version 4.x (and 5.x) support hot-plug operations, BBC, memory controllers, IOC, I^2C bus, Sun Fireplane bus, and other system operations and devices specific to the new servers.

The big difference in OPB 5.x is that it is written in Java and C, and is emulated in the SCApp environment on the system controllers as FCode. OPB version 1.x through 4.x are written using the FORTH programming language.

Refer to the OBP reference guides available at http://docs.sun.com for more info. Refer to Chapter 3 for more info on this OBP.

Sun Fire Configuration

Configuration of a Sun Fire demands considerable insight into the system hardware and desired functionality because of the options available. In this section, we describe commands for gathering configuration information and dynamically reconfiguring a system.

Configuration and Status Commands

There are a number of useful commands to gather configuration states on Sun Fire servers. When Solaris is up and running, you can use any of these:

/usr/platform/sun4u/sbin/prtdiag. This is the most useful command for most system administrators. This command will give you the server's architecture type, server model, clock frequency, total memory, keyswitch position, and detailed information about the CPUs, memory, I/O devices and, on the midrange and high-end servers, it will also include board states, available boards for the domain, and hardware failures.

TIP **Take advantage of the -v option for more verbose information.**

/usr/sbin/prtconf. This command gathers the system configuration information from a driver called devinfo (you can get more info on this by running the command *man devinfo(7d)*). Prtconf outputs information about the server's architecture type (all of these servers are the sun4u architecture), the total memory, and a breakdown of the device tree, and will tell you if the device drivers for each device is not loaded.

/usr/sbin/sysdef. This gives you much of the same info as prtconf, plus the hosted, swapfile information and loaded kernel modules.

eeprom. This command will give you info on the current OBP environment settings and allow them to be changed.

dmesg. This command displays a buffer of recent diagnostic messages.

format. This command displays hard drive information.

luxadm. This command displays FC-AL device information.

If the server is at the OPB ok prompt, you can use these commands to gather device status:

printenv. This displays the current OBP environment variable settings.

The probe commands (probe-scsi-all, probe-ide, probe-fcal-all). These commands display information about the various drive busses.

watch-net-all. This tests the network devices on the server.

If you are accessing the midrange or high-end server using the SC, you can run these commands:

showboards. Shows the assignment information and status of the boards.

showcomponent. Displays the blacklist status for a component.

showenvironment. Displays the environmental data (e.g., temperature and voltage).

testboard. Tests the board.

The Solaris Management Console and the Sun Management Center 3.0 software provide lots of configuration information using a GUI. Both products come with the Solaris Server kit. See Chapter 9 for more information on these tools.

Dynamic Reconfiguration

Dynamic reconfiguration allows you to change the configuration of a running system by bringing components online or taking them offline without disrupting system operation or requiring a system reboot. Dynamic reconfiguration is not an option on the Sun Fire 280R and V480 servers. On the V880, PCI cards can be dynamically reconfigured. On the other Sun Fire servers, system boards, I/O boards, and cPCI or hsPCI boards can be dynamically reconfigured.

Hot-pluggable devices can be safely added or removed from a system when the power is on, and require manual intervention (the administrator running specific commands) in order to dynamically reconfigure the component. Hot-swappable components can be removed automatically from OE control just by removing them.

WARNING Be aware that if the hot-swappable device is connected to a critical resource and is removed without checking to see that the resource is not controlling a critical resource (such as the boot drive), it could adversely affect operation of that resource. So, even with hot-swappable cards, it is smart to check first, using the appropriate commands.

In mission-critical environments, dynamic reconfiguration can be used if a system board fails and must be replaced or if new system boards need to be added to the system for additional performance and capacity. Dynamic reconfiguration capability is built into the Solaris Operating Environment with the Solaris 8 02/02 release and is a critical part of the concurrent maintenance strategy on Sun servers that support it.

Why can't all systems and all components use dynamic reconfiguration? Because all devices supported on that system must be *suspend-safe*. A suspend-safe device will not write to memory when the OE is in the *quiescence* state during the dynamic reconfiguration process. Quiescence is when the OE is paused to allow the copying of nonpageable (permanent) memory locations from the board being removed to another board. All operating environment activity sending information to memory, CPUs, and I/O devices must be halted during this critical period in the process. If an I/O card is not suspend-safe, it could write to memory and fail the dynamic reconfiguration process. All devices that can be used with dynamic reconfiguration need to be hot-pluggable.

On the midrange and high-end servers, memory interleaving across boards must be disabled, or dynamic reconfiguration will not work. When removing devices, you cannot remove critical resources from the system using dynamic reconfiguration. For example, you cannot remove the only system board with memory or CPUs, you cannot remove an I/O board or card connected to the boot drive.

NOTE If you are using a multipathing technology such as IPMP, MPxio, AP, or Veritas DMP, you must alert the software that a device under its control is to be removed using dynamic reconfiguration. See the documentation for the software for the correct procedure to do this.

All I/O activity on any I/O board or card to be removed needs to be stopped before removal using dynamic reconfiguration. This means all mounted file systems need to be unmounted and any volume management software needs to remove the associated drives from its control. Network devices would need to be brought down using the `ifconfig` command.

On the Sun Fire V880, the PCI cards require software intervention in order to remove or insert them into a fully operating system using dynamic reconfiguration.

This can be accomplished using any of these methods:

- Use the pushbuttons on the PCI Flex Circuit. The PCI Flex Circuit holds not only the diagnostic and status LEDs for the PCI cards, but also pushbuttons for readying the card to be removed from the system.
- Execute the `cfgadm` command. This command allows you to execute configuration administration operations on dynamic reconfiguration-compatible hardware resources. It is used to administer these cards for the dynamic reconfiguration procedure as well as to determine the status and layout of system cards. See Table 18.5 for `cfgadm` options.
- Use the Sun Management Center software.

WARNING No matter which method is used, the card should not be removed until the OK-to-Remove LED remains steadily lit.

Table 18.5 cfgadm Command Options

OPTIONS	DESCRIPTION
`-a`	Specifies that the -l option must also list dynamic attachment points.
`-c function`	Performs the state change function on the attachment point specified by `ap_id`. Specify `function` as `insert`, `remove`, `disconnect`, `connect`, `configure` or `unconfigure`.

(continues)

Table 18.5 cfgadm Command Options *(Continued)*

OPTIONS	DESCRIPTION
`-f`	Forces the specified action to occur.
`-h [ap_id |ap_type ...]`	Prints out the help message text. If `ap_id` or `ap_type` is specified, the help routine of the hardware specific library for the attachment point indicated by the argument is called.
`-l [ap_id | ap_type ...]`	Lists the state and condition of attachment points specified.
`-n`	Suppresses any interactive confirmation and assumes that the answer is no.
`-o hardware_options`	Supplies hardware-specific options to the main command option. The format and content of the hardware option string is completely hardware specific.
`-s listing_options`	Supplies listing options to the list (-l) command.
`-t`	Performs a test of one or more attachment points.
`-v`	Executes in verbose mode.
`-x hardware_function`	Performs hardware-specific functions.
`-y`	Suppresses any interactive confirmation and assumes that the answer is yes.

Here is an example of the `cfgadm` command run on a Sun Fire V880:

```
# cfgadm
Ap_Id          Type            Receptacle      Occupant      Condition
Sba            cpu/mem         connected       configured      ok
SBb            none            empty           unconfigured    ok
SBc            none            empty           unconfigured    ok
SBd            cpu/mem         connected       configured      ok
c0             scsi-bus        connected       configured      unknown
c4             scsi-bus        connected       configured      unknown
c5             scsi-bus        connected       unconfigured    unknown
pcisch0:hpc1_slot0  unknown    empty           unconfigured    unknown
pcisch0:hpc1_slot1  unknown    empty           unconfigured    unknown
pcisch0:hpc1_slot2  unknown    empty           unconfigured    unknown
pcisch0:hpc1_slot3  unknown    empty           unconfigured    unknown
pcisch2:hpc2_slot4  unknown    empty           unconfigured    unknown
pcisch2:hpc2_slot5  mult/hp    connected       configured      ok
pcisch2:hpc2_slot6  pci-pci/hp connected       configured      ok
pcisch3:hpc0_slot7  pci-pci/hp connected       configured      ok
pcisch3:hpc0_slot8  pci-pci/hp connected       configured      ok
#
```

Table 18.6 Device Types

TYPE	DESCRIPTION
mult/hp	Multifunction
Ethernet/hp	Gigabit Ethernet
pci_pci/hp	Quad Ethernet
scsi/hp	SCSI
raid/hp	Hardware RAID
tokenrg/hp	Token Ring
fddi/hp	FDDI
atm/hp	ATM
network/hp	Network interface (unspecified)
storage/hp	Storage interface (unspecified)
display/hp	PCI card (unspecified)
pci-card/hp	Graphics interface (unspecified)
unknown	Board/card cannot be determined
cpu/mem	CPU/memory board

The first column in the `cfgadm` output, `Ap_Id`, refers to the attachment point ID. `SBa-SBd` refers to CPU/memory slots A through D, `c0-c5` refers to the SCSI controllers, and `pcisch0:hpc1_slot0-pcisch3:hpc0_slot8` refers to PCI slots 0 through 8.

The second column, `Type`, designates the device type. The types are shown in Table 18.6.

The third column, `Receptacle`, designates the receptacle state. These states can be:

STATE	WHAT IT MEANS
empty	Slot is empty.
connected	Slot is electrically connected.
disconnected	Slot is not electrically connected.

The fourth column, `Occupant`, designates the occupant state. These states can be:

STATE	WHAT IT MEANS
configured	Logically attached to OS.
unconfigured	Logically detached from OS.

The fifth column, `Condition`, designates the condition of the device. That could be:

STATE	WHAT IT MEANS
ok	The device is ready for use.
unknown	The condition of the device cannot be determined.
failing	A problem has developed with the device.
failed	Failed component.
unusable	Incompatible hardware/empty, attachment point lacks power or precharge current.

To remove the PCI card from the Sun Fire V880 (this does not work for the 280R or the V480), use one of the three methods (command-line interface, pushbutton, or SunMC) to get the card ready for removal. Because of space limitations, we will only discuss the command-line interface method here. When using the `cfgadm` command, you do the following:

```
# cfgadm -c unconfigure pcisch2:hpc2_slot6
# cfgadm -v
< ... OUTPUT OMITTED ... >
Nov 21 15:15 unknown n /devices/pci@9,700000:hpc2_slot6
# cfgadm -c disconnect pcisch2:hpc2_slot6
# cfgadm -v
pcisch2:hpc2_slot6 disconnected unconfigured unknown
Nov 21 15:15 unknown n /devices/pci@9,700000: hpc2_slot6
#
```

Adding a PCI board requires the use of the `cfgadm - c` connect command:

```
# cfgadm -c connect pcisch2:hpc2_slot6
Nov 21 15:27:14 daktari-cte pcihp: NOTICE: pcihp (pcisch2): card is
powered on in the slot hpc2_slot6
```

You should verify that everything is okay by running the `cfgadm -v` command. See the SunMC documentation for the procedure to use SunMC and dynamic reconfiguration on the Sun Fire V880.

The procedure is similar on the midrange and high-end servers, using `cfgadm` from the domain. The high-end servers also have a SMS utility that allows you to initiate dynamic reconfiguration from the SC. This utility is called `rcfgadm`.

The `rcfgadm` utility allows you to:

- View the current configuration status
- Assign/unassign a system board to a domain
- Disconnect a system board from the OE to prepare it for removal
- Connect a system board for the OE to prepare it for removal
- Unconfigure a system board from the OE to prepare it for removal
- Configure a system board for use by the OE

If a domain is running, the `addboard` command on the SC can be used to dynamically reconfigure a resource into the running domain. The `deleteboard` command can be used to dynamic reconfiguration a resource out of a domain. The `moveboard` command will combine all functions, and remove a resource from a running domain and add it into another running domain.

You can also dynamically reconfigure the hsPCI cards by removing them. The Sun Management Center software can also be used on the midrange and high-end servers to dynamically reconfigure resources on the servers. The V880 only supports dynamic reconfiguration on the PCI cards.

Sun Fire Administration

Administering a Sun Fire server can be anything from an easy task to one that is quite daunting, depending on the type of server you are administering and how well you know the commands available to help with this task. Many of us have never before managed a system running multiple independent copies of Solaris, for example, and may have to adjust our thinking to accommodate the possibilities that this opportunity provides. In this section, we discuss administration commands for each of the server lines.

On the Low End—Workgroup Server Administration

Provided you have the minimum release of Solaris installed, these systems will run Solaris just like any of the older Sun servers. Only one independent copy of Solaris can run on these systems. Some of the useful admin commands for these systems were covered in the *Configuration and Status Commands* section earlier in this chapter.

Midrange Server Administration

The most important three things that administrators must master when first learning how to administer these systems:

- What command executes the task that you need to perform?
- Where do you need to be in order to run that command?
- How do you get there?

SCapp is a UNIX-like environment that runs on the system controller board. It has a CLI and provides console access. There are different administration shells that provide access to commands that perform various tasks. The *platform shell* provides access to commands that can affect the entire server. There are four *domain shells* (one for every possible domain) that provide access to commands that affect a specific domain. From within a domain shell, you can access the OBP environment after the domain runs POST, and if the operating environment is booted, we can access it through the domain shell as well. This provides administrators with access to the OE even when the primary network connection to the domain is not operating properly. The following sections discuss things to keep in mind when working with midrange servers, and show you some of the things that can be done in the platform shell.

Midrange Hardware Requirements

The midrange servers can support multiple independent copies of Solaris running at the same time. The Sun Fire 3800, Sun Fire 4800, and Sun Fire 4810 can support either one or two domains. The Sun Fire 6800 can support one to four domains.

Your system must meet the minimum hardware and software requirements in order for a domain to boot properly and operate. The minimum software requirements were listed in Table 18.2 earlier in the chapter. The minimum hardware requirements for a domain are as follows:

- A working and tested CPU on a system board assigned to the domain
- Working and tested memory located in a memory bank controlled by a working CPU on a system board assigned to the domain
- A working and tested I/O board assigned to the domain that is connected to a functional boot device with the minimum OE installed and configured properly
- Adequate power and cooling for all devices in the domain
- Compatible firmware on all devices in the domain
- Intercompatible hardware (for instance all boards in the domain must operate at the same clock speed)
- No overheated temperatures or hardware warnings (that were caused by failures)
- A required domain list that can be satisfied (in other words all boards listed to be in a domain are present)

If these requirements can be met, the domain can be booted.

Navigating between Shells

Navigating between the various shells seems to be one of the hardest things to get used to. The good news for administrators that support midrange and high-end Sun Fire servers is that the commands and procedures are the same or very similar to set up domains.

The SC Prompt

When you first log into the SC, you will see this prompt (provided the SC passed POST):

```
Type 0 for Platform Shell
Type 1 for domain A console
Type 2 for domain A console
Type 3 for domain A console
Type 4 for domain A console
Input:
```

If you want to go directly to the domain shell for any of the domains, type the appropriate number. If you want to connect to the platform shell, for example, you would type 0. You would then get a prompt containing the SC's name (which you assign to it when you first set up the SC) and the characters `SC: >`. For example:

```
sunfire2 SC:>
```

In this example, `sunfire2` is the name assigned to the SC. When you see the `SC: >` prompt, you know you are at the platform shell level. The commands that can be run on the SC can be listed by running the `help` command. For example:

```
sunfire2 SC:> help
```

All the commands supported by the particular version of the firmware will be listed.

NOTE We firmly suggest that you read the documentation that comes with the firmware (which can be downloaded from http://sunsolve.sun.com). That documentation contains all of the command syntax and in depth uses of these commands.

Navigation Commands

Navigation between the different administration shells relies on a number of different commands. To navigate from the platform shell to the domain shell, you use the `console` and `disconnect` commands. For example, the following three commands log you into the C domain (`console c`) and then disconnect (`disconnect`) you from the same domain:

```
sunfire2 SC: > console c
sunfire2 C: > disconnect
sunfire2 SC: >
```

If the domain has POSTed and you are at the OBP ok prompt, you can get back to the domain shell prompt by using the `break` command. For example, if you used Telnet to connect to the SC, hold the Control key down and at the same time press the] key.

```
ok <Control> ]
telnet> send break
sunfire2 C: >
```

If you used `tip` to connect to the SC, instead type ~# at the ok prompt:

```
ok   ~#
sunfire2 C: >
```

To return to the ok prompt use the `break` command:

```
sunfire2 C: > break
ok
```

If the OE is currently running, when you `console` into the domain, you will go to the login prompt. Log in as you would normally. If you want to navigate back to the domain shell type, send a break:

```
% <Control> ]
telnet> send break
sunfire2 C: >
```

(or use ~# if you are using a tip session). You can go back to the OE from here with the `resume` command:

```
sunfire2 C: > resume
%
```

The Domain Shell

Some of the other things you can do in the domain shell are:

- Set domain environment variables (using the `setupdomain` command)
- Configure the domain resources (using the `addboard`, and `deleteboard` commands)
- Monitor the domain environmentals (using the `showenvironment` command)
- Change the state of a domain (using the `setkeyswitch` command)
- Power on/off individual components in the domain (using the `poweron` and `poweroff` commands)
- Set/show the time/date/timezone of the domains (using the `setdate` or `showdate` commands)
- Reboot the SC (using the `reboot` command)
- Set password security on the domain shell (using the `password` command)
- Show the status of boards and components assigned to the domain (using the `showboards` and `showcomponent` commands)
- Test the boards in isolation (using the `testboard` command)
- View up to two days' worth of syslog logging stored in memory on the SC for the (using the `showlogs` command)

Configuring the SC

When first configuring the SC, you will notice that it will not have a network address and assigned name (unless Sun's support people set up the server for you). Without a network identity, you will not be able to Telnet to the SC. So, you will have to connect to the SC via the serial port. Once you log onto the SC, you will need to run the `showplatform` command to look at the current settings. If the network settings are not established or are incorrect, you can set them using the `setupplatform` command.

This command will ask you if:

- You are using DHCP or a static IP address. If static is chosen, the system will then ask for the net mask, default router (if used), DNS domain (if used), primary DNS server (if DNS is used), and secondary DNS server (if used).
- The log host (a system that the syslog system messages can be forwarded to).
- The platform description.
- The platform contact (text listing the name and contact info of the person responsible for the system).
- The platform location.
- SNMP settings. For example: Do you want to turn on the SNMP messaging system? Where would you like the SNMP messages to go? What SNMP security settings will you use?
- ACL settings (these control the ability to add or delete system boards or I/O boards from a domain).
- SC POST level (controls the level of POSTing done on the SC [off/min/max]).
- Partition mode (controls the functionality of the switch boards and how they provide Level 2 switching to the domains).

The firmware on the various boards on the server contains software code for the POST. This firmware can be updated by downloading the patch containing the firmware flash files from http://sunsolve.sun.com for the appropriate system. Uncompress the patch and use the `flashupdate` command. The detailed installation steps are contained in a PDF file that is included in the patch download.

NOTE It is important to stay current on these patches to support new features that are released periodically for these systems. For instance, you must have the 5.13 firmware (or newer) to enable the dynamic reconfiguration capabilities on these systems.

Setting up a New Domain

When the server is powered on, the current state of the keyswitch (which is stored on the SC in memory) and various domain environment variable settings determine if the domain is to be powered on, initialized, tested, and booted. The *keyswitch* for each domain is virtual, and its position is examined by the SC after the POST is run to determine whether the domain should be started up.

The steps to set up a new domain are:

1. Log into the Platform Shell and use the `showboards` command to determine what resources are in the server and if they are available to be used in the new domain. For example:

```
sunfire2 SC: > showboards
Slot     Grd Pwr   Component Type    State     Status     Domain
/N0/SB0  0   On    CPU Board         Active    Passed     A
/N0/SB2  0   Off   CPU Board         Available Not Tested Isolated
/N0/SB4  0   On    CPU Board         Active    Passed     A
/N0/IB6  0   On    PCI I/O Board     Active    Passed     A
/N0/IB8  0   Off   PCI I/O Board     Available Not Tested Isolated
```

In this example, system board 0 and 4 and I/O board 6 are being used in domain A. These boards can only be assigned to one domain. You cannot split up CPUs on the same board between multiple domains. Notice system board 2 and I/O board 8 are available. We can use these boards for our new domain.

2. The next step is to assign the boards to our domain using the `addboard` command. For example:

```
sunfire2 SC: > addboard -d b sb2 ib8
```

3. Now confirm that those boards are assigned to domain B with the `showboards` command:

```
sunfire2 SC: > showboards
Slot     Grd  Pwr   Component Type       State    Status     Domain
/N0/SB0  0    On    CPU       Board      Active   Passed     A
/N0/SB2  0    Off   CPU       Board      Assigned Not Tested B
/N0/SB4  0    On    CPU       Board      Active   Passed     A
/N0/IB6  0    On    PCI       I/O Board Active   Passed     A
/N0/IB8  0    Off   PCI       I/O Board Assigned Not Tested B
```

4. Using the `console` command, we can connect to the domain shell for domain B. For example:

```
sunfire2 SC: > console b
sunfire2 B: >
```

5. You can set a number of domain environment variables (using the `setupdomain` command described earlier in this chapter) to:

   - Control how you want the domain to POST when booting
   - Control error reporting to be used when booting
   - Control memory interleaving (which can speed up performance)
   - Control auto-booting and how the domain will function if the OE fails
   - Control the forwarding of syslog and SNMP message forwarding

6. To power on the domain set the virtual keyswitch to the on position. For example:

```
sunfire2 B: > setkeyswitch on
```

The SC will look at the resources assigned to the domain, power them on if they are not already powered on, and look at the domain environment variables to determine what level of POSTing needs to be run (we discussed what happens during POST in the *POST* section earlier in this chapter).

Configuring a Domain

To configure a domain you must log into the system controller board using either the serial port or the network port on the SC. There is only one account on the SC, and a password can be set to limit access to the platform shell.

The platform shell allows administrators to perform the following functions on the midrange servers:

- Configure the platformwide parameters (which are set using a number of platform shell commands such as the `setupplatform`, `setdate`, and `setdefaults` commands)
- Set up the network interface for the SCs (using the `setupplatform` command)
- Configure partitions and domains (using the `addboard`, `deleteboard`, `setupdomain`, and `setupplatform` commands)
- Monitor the platform environments (using the `showenvironment` command)
- Change the state of a domain (using the `setkeyswitch` command)
- Power on/off individual components (using the `poweron` and `poweroff` commands)
- Set/show the time/date/timezone of the SC and the individual domains (using the `setdate` or `showdate` commands)
- Reboot the SC (using the `reboot` command)
- Set password security on the platform or domain shells (using the `password` command)
- Show the status of boards, components on boards and the SC (using the `showboards`, `showcomponent`, and `showsc` commands)
- Test the boards in isolation (using the `testboard` command)
- View up to two days worth of `syslog` logging stored in memory on the SC for the domains and the SC (using the `showlogs` command)
- Navigate between the different administration levels (using the `console`, `disconnect`, `break`, and `resume` commands).

Terminating a Connection

One of the interesting issues in managing domains is the limitation on the number of simultaneous connections to a domain. Though you can have up to 12 concurrent connections (you can use the connections command to determine who is connected to the SC), only one user can be connected to a domain shell at a time.

Suppose someone logs on to domain A's shell and then goes to lunch and forgets to terminate his or her Telnet session. If you then try to connect to the domain A shell, you would get a `connection refused, domain busy` error. You can terminate the other person's session by doing the following:

First, log in to the platform shell and run the `connections` command. For example:

```
sunfire2 SC: > connections
ID   Hostname    Idle   Connected On     Connected To
1    localhost   01:04  Nov 23 11:45     Platform
2    10.1.2.5    00:03  Nov 23 12:14     Domain C
3    10.2.4.5    06:32  Nov 23 09:03     Domain B
7    10.1.2.34   56:35  Nov 23 11:06     Domain A
9    10.2.3.3    04:12  Nov 23 12:50     Platform
```

In this example, ID 7 is connected to the domain A shell and has been idle for over 56 minutes. You can terminate this connection by executing the following command:

```
sunfire2 SC: > disconnect 7
```

You will then be able to console into the domain A shell.

High-End Server Administration

The Sun Fire 12K and Sun Fire 15K provide remote administration capabilities like the midrange servers. You log into the SC using a regular Solaris account because the SCs are running a full installation of Solaris. Unlike the midrange servers, there is only one administration shell on these servers.

Management Networks

The system controller provides for administration of the high-end Sun Fire servers through a combination of hardware and software referred to as the management networks or MANs. The MANs should not be thought of as general-purpose networks nor used for anything beyond their intended purposes. The MANs provide critical services to the Sun Fire including:

- Domain consoles
- Message logging
- Dynamic reconfiguration
- System controller heartbeats
- Network boot and Solaris installation (optional)
- Time synchronization

The SC boards on the high-end servers each have twenty RIO chips that provide a 100 megabit (ERI) network port (one for each domain, one for the public network, and one for the private SC to SC network) as well as two 100 megabit (HME) network ports (one for the public network and one for the private SC to SC network).

In other words, there are three MAN networks on these machines, each which serves a different function. These are the I1, I2, and C networks. These three management networks are described below.

I1 Network. This network consists of 18 ERI ports (all internal-—no cabling) one to each domain connecting the SC to each I/O board in the server. This private network provides a highly secure console interface and allows for private data transfers on the server between the SC and the domains. Even though both SCs have an I1 connection to each possible domain, only the SC acting as the *main* (the one actually controlling the server—the other one acts as a *spare*) is connected to the I1 network. If the main SC fails, the spare SC's network ports are connected, maintaining the I1 network integrity. The I1 network also provides a mechanism for `syslog` messaging from the domains back to the SC. This network connection must be functioning properly in order for dynamic reconfiguration to work on the domain. The expander boards have an Ethernet hub to connect the I/O boards to both SCs. Each I/O board has a special IOSRAM ASIC that acts as a network port. This is used to make the domain side connection to the I1 network. If the domain has more than one I/O board, the lowest numbered I/O board is used to make the I1 network connection. Its IOSRAM is called the "Golden IOSRAM."

I2 Network. This network consists of one HME and ERI network port on each SC connected to its partner on the other SC. They provide a heartbeat and a private network used to synchronize the data on both SCs. This network is vital to allow the failover procedure to work properly.

C Network. This network consists of one HME (the upper RJ45 connector on the SC) and ERI network port (the lower RJ45 connector on the SC) on each SC connected to customers public network (or it can be connected to a secure private network if more security is needed. See the Sun Blueprints documents "Securing Sun Fire 12K and 15K Domains: Updated for SMS 1.2" and "Securing Sun Fire 12K and 15K System Controllers: Updated for SMS 1.2" at the following URL: www.sun.com/solutions/blueprints. If you run the `ifconfig -a` command on the system controller, the I1 network is represented by one logical connection called `scman0`. The I2 network connection is shown as `scman1`. The external network ports to the C network can be configured for higher availability by using *IP multipathing* (*IPMP*). When configured properly, IPMP will fail over to the other network port automatically. See the Sun Blueprints Document "IP Network Multipathing" at the preceding URL for more information on this topic.

System Management Services

The MANs are configured for the first time after the installation of the system controller's *System Management Services* (SMS) software. Usually the configuration of the MANs is done for you when Sun sets up your Sun Fire 12K or Sun Fire 15K.

The SMS software is installed at the factory. If you need to configure the MANs, run the `smsconfig -m` script. Without SMS, the two SCs would not be able to communicate and provide you with the needed services to configure and maintain the domains.

The configuration file created by `smsconfig -m` on the SC is located at: `/etc/opt/SUNWSMS/SMS1.x/config/MAN.cfg`. The logical and internal network settings are stored in this file.

If you needed to reinstall SMS, the SMS software is on the Solaris 8 7/01 release or newer supplemental CD. Fifteen packages are installed for SMS.

Once installed, SMS will be initialized through a series of scripts when Solaris starts up on the system controller. SMS provides a CLI interface to configure the MAN as well as to set up how you want it to behave when started.

SMS performs the following functions:

- Groups resources into *domain configurable units* (DCU). DCUs define what resources are used in a domain. (i.e. Expander, slot 0,and slot 1 boards) Each domain will run its own independent copy of Solaris.
- Runs the diagnostic programs on the server and specific domains as needed. It will automatically reboot and, if needed, reconfigure a domain if there is a serious hardware failure.
- Monitors and controls the power to the various components within the server. It also can warn you of a problem with power within the server.
- Facilitates the dynamic reconfiguration procedure to allow you to dynamically attach and remove resources in a domain. This capability can be automated using scripts.
- Interacts with the I^2C environmental monitoring sensors and reports the current temperatures, voltages, and amperages of the boards and ASICs in the server. It also can warn you of a problem with any component being monitored within the server.
- Provides administrative commands to allow domains to be booted and tested manually.
- Provides a higher level of security because it uses the authentication of Solaris to control what accounts have privileges to do specific SMS tasks. There are a number of special groups that are created by SMS that have specific privileges to run certain SMS commands.
- Keeps log files that cover the interaction between the domains and the SC. Log files are also kept covering what the users did in the CLI (this works like `netcon` on the Starfire).

SMS is started by the `/etc/rc2.d/S99sms` script. This script runs the `ssd_start` script, which in turn initializes the SMS Startup daemon (`ssd`) and the Message Logging daemon (`mld`). Up to 14 additional daemons are started on the main SC by `ssd`. These other daemons are responsible for the operation of SMS.

The `mld` daemon captures the output of all of the SMS daemons and associated processes. The behavior of `mld` can be configured in the `/var/opt/SUNWSMS/adm/.logger` file (only under special circumstances would this need to be done).

SMS Logging

The cumulative amount of log files stored on the SC is set to 550 MB. This can be set on the SC in the `/etc/opt/SUNWSMS/config/mld_tuning` file.

Following is a list of the locations for SMS log files:

- The log files for all platform (serverwide) messages are located at `/var/opt/SUNWSMS/adm/platform/messages.`
- The log files for each domain's messages are located at `/var/opt/SUNWSMS/adm/domain_id/messages. domain_id` will be the letter of the domain from A through R on the Sun Fire 15K and A through I on the Sun Fire 12K.
- The log files for the domain console messages are located at: `/var/opt/SUNWSMS/adm/domain_id/console`.
- The log files for the domain `syslog` messages are located at `/var/opt/SUNWSMS/adm/domain_id/syslog`.

Once they reach a specific size, the log files are renamed x.0 thru x.9 and, at most, 10 files of each type are stored in the appropriate location. Most of the binary files for SMS are located at `/var/opt/SUNWSMS/SMS1.x/bin`. The `ssd_start` script is located there as well.

Each of the domains has a unique MAC address and additional needed information that is installed at the factory. That information is stored in the `nvramdata` file. It is located along with other critical boot files in the `/var/opt/SUNWSMS/SMS1.x/data` directory.

Refer to the *System Management Services 1.x Installation Guide and Release Notes* for more information on the installation of SMS and refer to the *System Management Services 1.x Administration Guide* for more information on the SMS commands. Both of these documents can be downloaded from the http://docs.sun.com Web site.

SMS Accounts

The account used to log onto the SC can be given group membership into 39 special SMS groups that provide administrators with access to the various SMS commands:

- The Platform Administrator Group
- The Platform Service Group
- The Platform Operator Group
- Domain A through Domain R Administrator Group (18 groups)
- Domain A through Domain R Configuration Group (18 groups)

The SMS documentation can give you more details on which commands are available with membership of each group. The `/etc/group` file is used to maintain information on accounts in regard to secondary group membership.

The `smsconfig` command assigns a user to a group. The options available with this command are explained in Table 18.7.

Table 18.7 smsconfig Options

OPTION	FUNCTION
a	Assigns a user to a group
r	Removes a user from a group
l	Lists users in a group
u	Specifies the username (which must already exist on the SC)
G	Specifies the symbolic role of the user

The command synopsis for `smsconfig` with the a and r options looks like this:

```
smsconfig -a|-r -u username -G admn|oper|svc platform
smsconfig -a|-r -u username -G admn|rcfg domain_id
```

The values specified in these command forms are as follows:

admn. The platform or domain administrator group.

svc. The platform service group.

oper. The platform operator group.

rcfg. The domain reconfiguration group.

domain_id. The id for the domain (valid values are A through R and are case sensitive).

platform. Specifies the entire host machine.

Note that all of these values, except for *domain_id* are literals (to be entered exactly as shown).

For example, to assign user1 to the platform administrator group, use the following command:

```
sc0> smsconfig -a -u user1 -G admn platform
```

To assign user2 to domain F's administrator group, use the following command:

```
sc0> smsconfig -a -u user2 -G admn F
```

If a user has platform administrator capabilities, he or she can:

- Restart the system controllers
- Configure platformwide parameters
- Display platform configuration information

- Monitor system environmental information
- Power on/off the system components
- Set the time-of-day clock
- Back up and restore the SMS environment
- Update the system firmware

The *System Management Services 1.x Reference Guide* (available at http://sun.docs.com) contains more information about these commands. Many of the commands are similar to the previous commands discussed earlier for the midrange servers.

Remote Administration

All of the Sun Fire servers can be administered through the console bus by connecting a local ASCII device such as an ASCII terminal, a laptop, a network terminal server, a PDA, or another Sun system using a null modem cable to the server's external serial port and the `tip` command.

Most administrators would not want to work inside a room full of servers all of the time. These rooms can be very noisy. Instead, they can use the remote administration technologies built into the Sun Fire servers to access the console from another location.

All three of the Sun Fire workgroup servers use the *Remote System Control* (RSC) board and RSC software to facilitate remote administration.

All four of the Sun Fire midrange servers use the *System Controller* (SC) board and SCApp located in firmware to facilitate remote administration.

Both of the Sun Fire high-end servers use the *System Controller* (SC) board and SMS software to facilitate remote administration. Let's look at how this works on the different servers.

Remote Administration for Workgroup Servers

The RSC board is installed in a special slot in each of the three Sun Fire workgroup servers. This special slot is keyed differently from the PCI card slots.

The RSC hardware and software offers the following features on these servers:

- Provides remote access to the host system console—you can even log the boot activity.
- Performs a hard reset, also known as a Power On Reset (POR)—this feature is useful when no one is present to reset the key switch. You can also initiate a soft reset, also known as an Externally Initiated Reset (XIR)—this feature is used by Sun support to determine what state the hardware was in during a failure.
- Monitors the host environment and sends alerts for critical events—this can be in the form of email or a page.
- Provides the ability to power on, power off, and reset the host.
- Provides remote "lights out" management of the host system—there is a battery pack (3cell, LL_AAA, or 3.6V.) to operate the RSC card when the facility power goes out. It takes 24 hours to recharge the battery pack.

- Displays environmental information—the card and software can interact with the I^2C bus.
- Provides built-in security: four individual username and password accounts with configurable access rights, login tracking, PPP authentication, and notification upon trying to breach RSC security measures.

The RSC card contains a chip that holds the RSC firmware. It contains configuration settings for the console ports on the RSC, and it enables you to create one initial RSC user account.

The RSC 2.2 software is on the Solaris supplemental CD, and you will need these minimum software requirements on the servers:

- Sun Fire 280R server running the Solaris 8 (1/01) OE or later
- Sun Fire V480 server running the Solaris 8 (02/02) OE or later
- Sun Fire V880 server running the Solaris 8 (7/01) OE or later

You can also download the RSC software from www.sun.com/servers/rsc.html.

There are two components to the RSC 2.2 software:

- RSC 2.2 server component
- RSC 2.2 client component

The client component is installed on the remote system that you want to use to communicate with the RSC card/Sun Fire workgroup server.

After installation, the RSC server software will reside in the location shown in Table 18.8.

The RSC 2.2 software provides you with a CLI interface or a GUI. There are three user interfaces for RSC:

- The RSC shell
- The `rscadm` utility
- The client GUI

Table 18.8 Resource System Control (RSC) Software Locations

SUN FIRE MIDRANGE SERVER	RSC SOFTWARE LOCATION
Sun Fire 280R	`/usr/platform/SUNW,Sun-Fire-280R/`
Sun Fire V480	`/usr/platform/SUNW,Sun-Fire-480R/`
Sun Fire V880	`/usr/platform/SUNW,Sun-Fire-880/`

The RSC shell and the GUI software do not require root permission on the RSC server system and can perform identical tasks. The `rscadm` command is located in the `/usr/platform/SUNW/Sun-Fire-880/rsc` directory on the RSC server machine. You can set up as many as four user accounts to access the RSC card. You must use the `rscadm` command to set up the first account. For example:

```
rsc> useradd user1
rsc> userperm user1 cuar
rsc> usershow
```

The `cuar` option when using the `userperm` command sets the account to access all of the commands available in the RSC software.

To start up the RSC GUI to connect to the RSC server software, you must run the following command:

```
# /opt/rsc/bin/rsc
```

The GUI window will appear. You need to know what server with a RSC card you would like to connect to and provide it with a username and password.

The main screen allows the user to access many functions, including server status and control, setting the bootmode, and viewing logs.

The RSC can be configured to change the configuration of the RSC serial port, Ethernet port, console access, alarm settings, notification method, and content alert variables.

The firmware on an RSC card firmware can be downloaded to the card using the `rscadm` utility's subcommand download (providing the keyswitch is *not* in the lock position).

```
# rscadm download [boot] file
```

You can designate the source file to download and the destination with these options:

- **(No boot):** Downloads the file to the main firmware section of RSCs nonvolatile memory.
- **(boot):** Downloads file to the boot section of RSCs nonvolatile memory.

After the firmware is installed on the card, the RSC will reset itself.

Remote Administration for Midrange and High-End Servers

On the Sun Fire midrange servers, console access can be provided using the serial port or the network port on the SC board.

The system controller software (SCApp) does the following:

- Monitors and controls the system
- Manages hardware

- Provides a platform console and a domain console
- Configures domains
- Provides the clock signal used on all system boards
- Provides the date and time to the Solaris operating environment
- Provides system monitoring and control using SNMP for use with the Sun Management Center 3.0 software

On the Sun Fire high-end servers, console access can be provided using the serial port or the network port on the SC board as well. The SMS software provides administration capabilities (see the *System Management Services* section earlier in this chapter. Refer to Table 18.3 for internal storage options for these servers.

Troubleshooting Tools for Sun Fire Servers

There are a number of tools available from Sun to help you troubleshoot these servers, including:

Sun Management Center. SunMC has a Hardware Diagnostic Suite module to run nondestructive tests on components in a running domain. It also has extensive logging and notification capabilities.

Sun Visual Test Suite (SunVTS). This software is designed to stress-test hardware and is *not* recommended for production systems. Back up the data and take the server offline before running this software! It is an excellent tool for catching hardware that works most of the time but is occasionally "flaking out." Be sure to only use the version of SunVTS that was released for the version of Solaris you have installed, or you might get some bogus responses.

Syslog logging. All of the servers can log hardware and software errors via the `syslogd` logging process. By default, these messages get logged in `/var/adm/messages` and are a good place to start when looking for problems.

POST. POST can report errors when identified during the initial testing of the system. Set the POST settings for maximum to get more detailed information.

Sun Explorer. This software is available for free to download on the `http://sunsolve.sun.com` Web site. Sun Explorer can take a "snapshot" of your current system configuration to aid Sun's technical support group in troubleshooting by collecting information on your current configuration.

http://sunsolve.sun.com. This Web site provides hardware specifications, system views and components, component lists, alerts, symptom resolution information, and much more.

PatchPro. This interactive (or automated) tool can also be downloaded from the Sunsolve Web site. It can scan your system to determine if you have the current supported patches installed. A majority of all problems on these servers can be corrected with the current supported patches.

Support documentation. Sun provides a ton of free documentation on their Web site. To obtain related support documentation, go to the sites shown in Table 18.9.

Table 18.9 Sun Fire Server Support Documentation

SUN FIRE SERVER	URL
V480	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire280R/docs.html`
V880	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire880/docs.html`
3800	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire3800/docs.html`
4800 & 4810	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire48x0/docs.html`
6800	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire6800/docs.html`
12K & 15K	`http://sunsolve.sun.com/handbook_pub/Systems/SunFire15K/docs.html`

Summary

In this chapter. we explained that Sun's new family of Sun Fire servers contain much of the same hardware as previous Sun servers, yet offer an astounding level of improvements over the earlier servers. They also provide a significantly higher level of RAS (reliability, availability, and serviceability)—built into the servers' basic architecture—than just about any other systems available today in the comparative price points.

We also saw the performance increase provided by the new Sun Fireplane Inter-Connect and its support ASICs. We saw the functional goals of each of the servers and their key features provided by these new technologies.

We provided a lot of information on how these servers are managed, while pointing out many additional documents available to you from Sun's Web sites. Should you have an opportunity to administer one or more of these magnificent machines, we hope that this information will give you a head start and encourage you to become an expert at managing the resources of your Sun Fire servers.

PART

IV

Surviving in the Real World

Today's networks are complex. Because they're connected to the Internet, the complexities and the problems are not all within your control. Some of the problems you'll run into will happen because of something out there affecting something within your organization. Not only will your users want to access services and systems outside of your organization; people outside of your organization will want to access services and systems inside your organization—both legitimately and not so.

Keeping your systems manageable in the context of these demands involves several critical steps: understanding the configuration of network services—especially those provided to the outside world—configuring services properly so that they provide the intended service and nothing more, and monitoring services so that you are able to detect when a problem is developing.

Of course, the difficulties are not all at the boundaries between your organization and the rest of the Internet. You may have many difficulties and, in fact, many boundaries within your organization if your systems and your users cannot work together effectively. The reality of today's networks is that they are increasingly heterogeneous. A network with Unix, all flavors of Windows, Linux, and Macintosh systems interspersed randomly in roles as servers and clients is not at all uncommon. In fact, some version of this mixed network is probably the norm. How you manage system services will determine to a large extent whether you and your users are constantly bumping up against problems sharing data, files, printers, and applications.

This *is* the real world. There is no escaping the Internet or the attraction of desktop systems with increasing power and decreasing price tags. Then again, we wouldn't *want* to. Along with the challenges of surviving in today's complex computing environments, there is a lot of excitement in the process of configuring and managing our networks. Even after as many years as we have spent doing Unix systems administration, we still get a thrill the first time some device or network service we're setting up suddenly springs to life. Today's networks are complicated and, at times, frustrating. Today's networks are fun. We really wouldn't have it any other way.

In Chapter 19, we talk about the problems and responsibilities associated with running an Internet site. In Chapter 20, we provide suggestions and techniques for coexisting with other empires—in other words, managing the heterogeneous network. In Chapter 21, we offer our best strategies not for managing your systems, but for managing your time, in the hope that the demands of your job don't overwhelm you. We hope that something within each of these three chapters will help you cope with issues in your real world.

CHAPTER

19

Running an Internet Site

What does it mean to run an Internet site? Most of us are on the Internet to a considerable degree. We get mail from around the globe. We search for technical answers and replacement hardware using our Web browsers. Running an Internet site is more than being *on the Internet*, however. It's *participating* on the Internet, as a source for tools, expertise, or other services.

This is true for businesses small or large, whether technical or not. Restaurants post their menus on the Web. Artists display collections of their works. Poets, sailors, architects, and accountants—all manner of craftsmen are offering their works for purchase or admiration on the Internet. Not just a population explosion, however, the trend toward near-universal Internet participation is also giving rise to new organizational structures. Increasingly, corporations are forming partnerships and taking advantage of the ease with which information can be shared and distributed. Corresponding changes, both technical and political, are needed to manage the delicate balances of competition and cooperation, threat, and opportunity.

Along with our participation on the Internet comes a new set of responsibilities and concerns. Security is one of the top concerns. How do we protect our assets from malicious and inadvertent damage? Liability is another. How do we keep our systems and our users from becoming active or unwitting participants in fraudulent activities? How do we ensure that our systems don't become repositories of unauthorized copyrighted materials? How do we avoid becoming conduits for spam? In this chapter, we offer some guidance with respect to these concerns and suggest some avenues for further exploration of security issues in Chapter 13.

The Dangers of an Internet Presence

There are several reasons why connection to the Internet is inherently dangerous:

- TCP/IP services are flawed.
- The Internet is growing every minute.
- Hackers are highly motivated.

Initially designed to be open, services that run on and run the Internet are now being retrofitted with restrictions and security extensions. With every security enhancement, there is an increased sophistication (read: *complexity*) and, as is often the case, a new and different set of problems. In addition, Internet connections are based on protocols that break network connections (whether connection-oriented or connectionless) into packets that travel freely across whatever routes seem best at the time. Unless precautions are taken to obfuscate their contents (i.e., by using encryption), they can be intercepted and, possibly, altered en route to their destination.

At the same time that we are facing these old and new dangers, use of the Internet is increasing faster than the Earth's ozone layer is diminishing. This steady rise of netizens brings additional problems—both advertent and inadvertent. Increased traffic leads to potential bottlenecks, service overruns, and increasing activity at Internet sites. There is also a growing population of hackers; most of them grew up on the Web and now use the Internet as a kind of cyberplayground for their malevolent hobby.

Many network services could be secured, at least in theory, but they cannot necessarily be secured in isolation (i.e., without consideration of other network services) or across the board on a highly heterogeneous network or the Internet. The most practical solution for many organizations is, therefore, to invest their efforts in single connections to the Internet that are both highly constrained and not very interesting.

Fortunately, tools for limiting the risk of participating on the Internet are more or less keeping pace with the dangers. Firewall technology and security-hardened network services are helping sysadmins to maintain some semblance of security on their networks. Office workers are dialing in to their ISPs and then using VPN products to securely connect to their offices. `sendmail` is refusing to provide lists of email addresses or to relay messages for strangers. Encrypting messages in most client email products is now as simple as clicking a button.

In addition, considerable Internet security expertise is available for those with the time to do a little browsing. We highly recommend sites such as www.cert.org (a center for Internet security expertise operated by Carnegie Mellon University) and www.honeynet.org (a nonprofit research group studying the tools, tactics, and motives of the hacker community) for keeping yourself informed.

The DMZ

The term *DMZ* stands for *demilitarized zone* and derives from international conflicts where a neutral buffer is established between warring parties. In a networking context, it's a barrier between your network and the outside world. A DMZ is not part of your

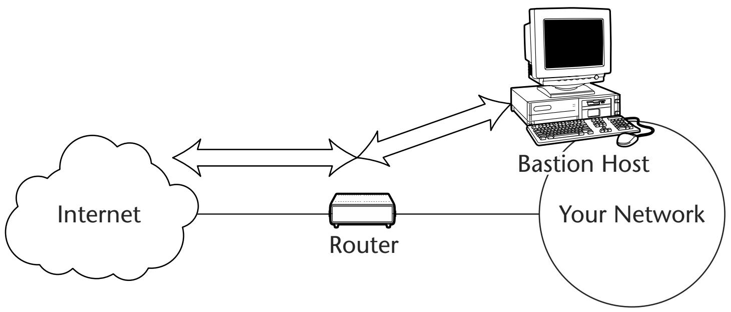


Figure 19.1 A bastion host.

network in the strictest sense of the word, nor part of the Internet. By putting what is essentially an informational airlock between critical resources and a myriad of unknown dangers, you reduce the risk of anyone on the outside taking advantage of your systems.

A DMZ is, typically, the networking between your Internet router and your bastion host. This can be accomplished in one of two fashions. Either the DMZ can be the segment between two physical firewalls, or it can be a third network interface off of your existing firewall. In either case, the normal access controls allow traffic to flow from the Internet to the DMZ, and from the DMZ to the internal network, but never would anything be allowed to flow directly from the Internet to your internal network. Figure 19.1 depicts a typical configuration in which a bastion host is deployed to restrict access to a network. The router only permits traffic to and from the bastion host.

Firewalls

Firewalls are like traffic cops. They look at all connections that try to make it across the barrier and decide whether to let them through, based on protocols or addresses. The word firewall, of course, derives from firewalls in other contexts—between you and the engine in your car or your townhouse and the one next door. The idea is similar; a firewall is a barrier against danger. On a network, a firewall can be a device unto itself, but is usually software running on a router or on a single-purpose host. It is unwise to use a firewall for any other purpose. It is also unwise to allow logins (other than those that are needed for its maintenance) or provide any additional services from a firewall-centric device for the simple reason that these services might be exploited and could provide ways around the firewall's constraints.

Firewalls are located at boundaries between organizations that do not want any and all traffic passing between them. Firewalls can be configured to pass or block data according to port, IP address, authenticated user, and other criteria.

WARNING The more sophisticated you make your firewall implementation, the more chances there are for error. The best policy is to start with a blanket "Deny all traffic" statement, then open up specific external hosts and ports for access to specific DMZ hosts and ports, and do the same for access between the DMZ and the internal network.

Decisions regarding the purchase and configuration of firewalls involve a number of factors. These include your basic goal, your organizational policies, your staff, and your budget. Whether you are trying to provide remote access service to a handful of roaming employees or ward off service attacks determines the solution you should choose. Without strong security policies and control over your organization's networking architecture, a firewall may be next to useless. You'll need to gauge the extent to which you can ensure that connections around the firewall are not permitted. You also have to take a hard look at your budget. Most commercial firewall packages will likely cost upwards of $100,000. Solaris 9 includes one for free, called SunScreen. We introduce SunScreen later in this chapter.

Firewalls provide these basic functions:

- A single control point for Internet security.
- Protocol filtering—blocking protocols that are not needed, especially those that are risky.
- Source/destination filtering—blocking sites you don't trust (or allowing *only* those you do).
- Information hiding—making information that is readily available inside your organization (e.g., usernames and host names) unavailable from the outside or on the firewall itself.
- Application gateways—stop points through which services users must connect before proceeding further, providing opportunities for extensive logging and application-level control.
- Concentrated and extended logging of activity across the firewall.

Firewalls cannot protect against every possible type of damage or attack. In fact, the worst firewall is one that's improperly configured. One of the biggest misconceptions out there is the thought that just because you have a firewall, you are safe. Most attacks are not against firewalls; rather, they go through the firewalls on ports that are already open. Following are some of the things that firewalls can't protect you from:

- Data walking out the door in someone's breast pocket
- Break-ins if someone leaves a modem hooked up to a workstation
- Compromises if firewalls are not configured properly
- Someone knowing an authorized user's username and password (and remote login is allowed)
- Viruses (except when firewalls are specially configured to do so)

In the following sections, we describe the basic types of firewalls and mention some alternatives.

NOTE **In spite of the simplicity of the filtering examples provided in the following sections, do *not* get the impression that configuring and maintaining a firewall is easy. There is a lot of thinking and testing needed before you'll be ready to establish the seemingly simple rules and meet the continuing need for monitoring and updating.**

Understanding Ports and Protocols

Before we begin discussing how firewalls work, a review of the basics of the Transmission Control Protocol/Internet Protocol (TCP/IP) and the role of ports and protocols is in order. TCP/IP is used on local area networks (LANs) as well as on the Internet itself. You don't have to know much detail about the internals of TCP/IP in order to use it, but any knowledge you have will undoubtedly come in handy when you run into problems.

TCP/IP refers to a family of related protocols. The specifications are published so that anyone can build compatible tools and products. The major protocols composing the protocol suite are the Internet Protocol (IP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

IP is the foundation for the other protocols and provides addressing—the familiar IP addresses in dotted octet notation (e.g., 192.90.200.2) that identify each host. IP provides addresses (IP or network addresses—*not* hardware or MAC addresses), but doesn't guarantee delivery of packets from one host to another. It leaves that part of the work to other protocols.

TCP provides for guaranteed packet delivery. This doesn't actually mean that packets always get where they're supposed to go, of course. Systems crash, routers fail, and problems with the receiving applications can result in data failing to reach the far end of a connection. Instead, it means that there is a mechanism built into the protocol for acknowledgment and for retransmission. Packets frequently don't reach their destination. Were these mechanisms not in place, few of our connections would seem reliable from our vantage point as sysadmins and as users.

TCP is called a *connection-oriented* protocol. This means that there are three phases to a TCP connection—setup, data transmission, and shutdown. In addition, packets are serialized and can be reordered if they arrive at the far end out of sequence. TCP also incorporates an important addition to the addressing structure of packets: *ports.* Ports identify particular services on the communicating systems that are active in the connection.

UDP is a *connectionless* protocol. There are no setup and shutdown phases involved in UDP transmissions. UDP is used by services that do not require anything more than the transmission of data. The common analogy is to say that TCP is similar in character to a phone conversation, while UDP is similar in character to a letter that arrives in the mail.

TCP/IP involves other protocols, such as ICMP, which we will not discuss here. There are many extremely good books written on the subject, and we encourage you to do further reading if you are interested in understanding the Internet extremely well.

Services

Most Unix servers support a wide range of network services, including FTP, Telnet, `sendmail`, POP, printing services, news and Web servers, and more. Each of these runs (i.e., listens for requests) on its own port. Some use UDP and others TCP; some services use both.

There are two basic ways that these processes are initiated—they start up as background tasks in the start/stop scripts (see Chapter 4) or they start up as requested through the services of `inetd`. To determine which of the processes on your systems start during bootup and changes of run state, examine the `/etc/init.d` and `/etc/rc?.d` directories. These include the `inetd` service itself, NFS, and `sendmail` and `httpd` processes, along with numerous others. Though many services could be initiated either way, there is generally a good reason for the choices made. For some services, such as `inetd`, this is clearly the only option. For others, it's only practical for them to run constantly in the background. Services that run through the auspices of `inetd` are more likely to be run *infrequently.* In addition, the services are often simpler, since some of the overhead involved in making and breaking connections is managed by `inetd` rather than by the services themselves.

The inetd Process

The `inetd` process is sometimes referred to as the *superserver.* It listens on many ports and starts the appropriate service as needed. To determine which of your services runs through `inetd`, you have only to examine your `/etc/inetd.conf` file. This is `inetd`'s configuration file. Here are a few lines excerpted from a typical `inetd.conf` file:

```
# <service_name> <socket_type> <proto> <flags> <user> <server_pathname>
<args>
# FTP and Telnet are standard Internet services.
#
ftp    stream  tcp  nowait   root    /usr/sbin/in.ftpd     in.ftpd
telnet stream  tcp  nowait   root    /usr/sbin/in.telnetd  in.telnetd
```

Like most files on Unix systems, comment lines start with a pound sign (#). The seven fields that compose each service line are as follows:

Service_name. As specified in the `/etc/services` file, which associates services with particular ports.

Socket_type. The type *stream* is used with TCP connections, and *dgram* (meaning datagram) is used for UDP.

Protocol. The protocol used by the server. This is almost always TCP or UDP, but you will see some RPC types as well.

Wait/nowait. This is a flag used to determine whether `inetd` will resume listening immediately (`nowait`) or only after the service terminates (`wait`).

User. The username the process runs under.

Service_pathname. The full path name to the service.

Args. The service's arguments. This includes the service name (and often no additional arguments).

Ports also fall into three broad categories—*well-known* (also called *reserved*), *registered,* and *ephemeral* (the rest). Ports with numbers below 1024 should only be used for the service for which these ports are reserved. If you ran `sendmail` on any other port than 25, after all, no other system would know how to communicate with it. Ports above 1023 are often used without concern for their official registrations. For example, you will often see user Web sites running on port 8000 (root's port is 80). Ports above 49151 are officially available for any service you might want to build or configure to run on them.

Now, let's briefly run through a typical connection. A user starts a Telnet session with a `telnet spoon` command. The user's Telnet process is allocated an unprivileged port on the local machine. That process connects with the privileged port on the system, spoon, the user is trying to reach (Telnet uses port 23). The remote Telnet daemon, `telnetd`, then responds to the assigned port on the client. The connection is established, and the Telnet session proceeds to completion. If you wanted to block Telnet sessions on spoon, therefore, all you would need to do is inactivate port 23. That's the only port that Telnet is listening on, even though the originating (client) process will be on some random port above 1023. This snoop excerpt shows the ports in an active Telnet session (from the server side):

```
spoon# snoop port 23
. . .
TCP: Source port = 23
TCP: Destination port = 1026
```

Some services do not use privileged (i.e., known) ports and are therefore harder to block. Your firewall may make it easy to block all ports with port numbers below 1024 (i.e., all privileged ports), but this will block all services. A combination of rules and exceptions may be what it takes to select the proper set of services.

Poor firewall management can be worse than having no firewall at all. Make sure that you get a firewall product that is easy to configure or that you spend enough time to fully understand what you are blocking and what you are allowing through.

Packet Filtering

The primary function of a firewall is to filter packets—basically, allowing them through or stopping them at the firewall. Packets can be filtered according to a number of criteria—protocol, IP address of the source or destination, or port (TCP/UDP) of the destination or source. If you want to prevent `ping` attacks, for example, blocking ICMP traffic at the firewall prevents these requests from reaching your hosts. This is very similar to the functions performed by access lists on routers.

The decision as to what protocols you want to filter depends on the particular policies your organization has in effect and what you are trying to accomplish. Typically, protocols that are blocked at the firewall include FTP, TFTP, X, RPC (including NFS, NIS, and

NIS+), r commands (e.g., `rlogin`), Telnet, RIP, and UUCP. If you're running a Web site, you'll need to support HTTPD and maybe also FTP. You probably need to allow email through, using SMTP. We suggest that you spend some quality time with your network services files (e.g., `/etc/services`) and determine what services you must support and which you can filter without affecting the productivity of your organization.

Packet filtering rules take a general form including packet type, source and destination addresses, source and destination ports, and what is to be done with packets that match the specified criteria. For example, a rule that looks like Table 19.1 would reject requests for any privileged port, regardless of where they were derived from. The network designator 0.0.0.0/0 matches *any* address. An address like 220.203.27.0/24 would match any host on the 220.203.27.0 network; the 24 indicates that the first 24 bits are significant with respect to the matching. The rule shown in Table 19.2 allows Telnet from one particular class C network.

The more sophisticated packet-filtering firewalls also allow you to configure when filtering happens—when a packet is received or when it is sent. These options are usually referred to by the terms *on input* and *on output*. They can also filter packets leaving your network as easily as those that are arriving; not every danger is out there. Other features include filtering, the ability to avoid the use of routes learned from other systems, and the ability to disable source routing. Firewalls that cannot be reprogrammed remotely provide an additional level of security; the last thing you want is for your firewall to fall into the hands of a hacker.

Packet-filtering firewalls are also called *network-level* firewalls.

Proxy Servers

A proxy server (also called an *application gateway*) is an application that sits between a network service and the Internet. Because it must speak and understand whatever protocol the application uses, it also can manipulate the data and commands that it intercepts. Most proxies, therefore, can exert finely tuned control over whatever service they are proxying for. They also provide fairly extensive logging and can add more rigorous authentication support than is generally enforced (e.g., they could authenticate themselves with another system before making the connection to the target service). They can also replace network addresses as they pass packets to the application and to the caller.

Table 19.1 Sample Deny Rule

FIELD	SAMPLE VALUE
Type	`tcp`
Source IP address	`0.0.0.0/0`
Destination IP address	`0.0.0.0/0`
Source port	`any`
Destination port	`< 1024`
Action	`deny`

Table 19.2 Sample Permit Rule

FIELD	SAMPLE VALUE
Type	`tcp`
Source IP address	`220.203.27.0/24`
Destination IP address	`0.0.0.0/0`
Source port	`any`
Destination port	`23`
Action	`permit`

Proxies are tied to specific network services. You might set up a proxy for Telnet, FTP, HTTP, and so on. A set of proxy servers, the TIS Internet Firewall Toolkit, is available from www.tis.com. There is also a generic proxy system called Socks that can be used to make a client-side application work through a firewall. Information on Socks is available from www.socks.nec.com.

When you use proxy servers, no network traffic goes directly to the services in question. Some proxy systems are less transparent to end users than their packet-filtering counterparts, especially older systems. Application-level firewalls permit no traffic directly between networks but instead run proxy servers.

Because these types of firewalls run processes to handle each application or port range separately, CPU usage and scalability is an issue. Make sure you define your requirements adequately prior to sizing a machine as an application firewall.

Stateful Inspection

Many modern network-level firewalls now maintain internal information about the state of connections passing through them. Similar to proxies in the level of application control available, firewalls equipped to do stateful inspection, examining the contents of data streams without requiring a proxy for each service to be monitored and without the performance hit that is easy to imagine might be possible. FireWall-1 and VPN-1 Gateway, both products of Check Point, include a Security Policy Editor that allows you to configure stateful inspection options.

Patented by Check Point, stateful inspection allows content-level inspection of traffic crossing a firewall. For example, you can:

- Check for malicious code (e.g., applets).
- Scan email attachments for code or viruses.
- Use built-in APIs to integrate third-party screening and antivirus applications.
- Ignore other than basic SMTP commands (hiding information and preventing SITE EXEC commands).

- Remove malicious code and viruses before forwarding the contents of the communication.
- Scan URLs for specific pages or content types—using wildcard specifications, file specifications, or third-party databases.
- Strip Java and ActiveX code and tags (protecting against attacks).
- Block spam relays.
- Block suspicious addresses.
- Hide internal addresses.
- Drop mail (i.e., filter) from given addresses.
- Strip attachments of given types from mail.
- Strip the received information from outgoing communications (concealing organizational information).
- Drop mail messages larger than a specified size.
- Run filename and virus checking for FTP.
- Implement optional blocking of FTP `get` requests.

Firewall Alternatives

Firewalls have also been constructed using *dual-homed hosts*—systems with two network interfaces, one connected to the outside and one to the inside network. With IP forwarding disabled, users would have to log in to the dual-homed system before proceeding past it. The system would then act as a choke point. Systems like these were more popular before the advent of the Web. Figure 19.2 illustrates the position of a dual-homed host. Figures 19.3 and 19.4 illustrate two additional firewall techniques—a screened subnet and a screening firewall. In a screened subnet, two routers are involved: one restricts access to the screened (outer) subnet, while the other passes traffic between the screened and the protected subnet. In the screening, or packet filtering, firewall, only certain protocols are allowed to pass.

Another alternative to a standard firewall is software, such as TCP Wrapper (described in Chapter 12), which adds access control to standard network services that are invoked through `inetd` (e.g., Telnet and FTP), but does not protect other services.

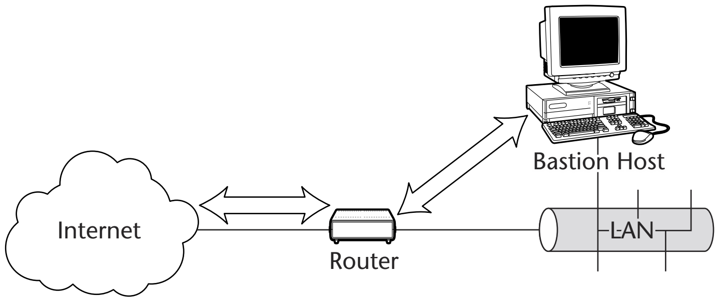


Figure 19.2 A dual-homed host.

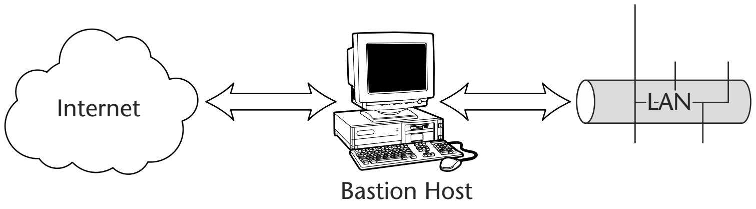


Figure 19.3 A screened subnet.

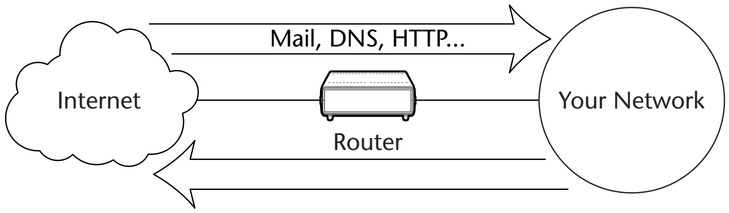


Figure 19.4 A screening firewall.

Firewall Security

For a firewall to provide security, it must itself be secure. Here are some general rules of thumb to help you keep your firewall functioning well and safe from tampering.

- Maintaining extensive logs on a firewall is important to help determine if the firewall is indeed working and to detect suspicious activity (which could be a precursor to an attack of some kind). Review these logs periodically.
- Do not assume the integrity of your firewall. Examine it from time to time.
- Install patches as required. (Don't think your firewall can't possibly need them.)
- Make sure your firewall is configured with adequate memory, and be attuned to any sign of hardware failure.
- Provide a backup strategy should your firewall fail.
- Limit or ban network logins to your firewall. If you must log in, use Secure Shell (SSH) or a similar encrypted method of access. Do not use Telnet, as your entire session can be sniffed and your IDs and passwords determined.
- Use authentication tokens (e.g., SecurID) if you must log in over the Net.
- Do not run any services on the firewall that don't relate to and are not needed by the firewall—no NIS, no NFS, and so on.

- Disable `finger`.
- Don't allow host trust (`~/.rhosts` or `hosts.equiv` type access) on your firewall.

SunScreen 3.2

SunScreen is a serious and versatile firewall product that is provided free of charge with Solaris 9. Integrated into the operating environment, SunScreen can provide a far more significant function than protection of the perimeter of the enterprise: it can be used throughout the enterprise to provide access control, authentication, and data encryption.

SunScreen provides a rules-based, stateful packet-filtering engine that can be used to create virtual private networks (VPNs) by adding public-key encryption technology to the network. It is also one of the first firewall solutions to incorporate the concept of high availability (HA) into its design. SunScreen includes the following features:

- Stateful packet inspection (also referred to as *stateful packet filtering*, as described earlier in this chapter)
- Stealth and routing firewall features
- Standalone IPsec/IKE capabilities
- Centralized management
- Failover
- Proxy services for Telnet, FTP, HTTP, and SMTP (with antivirus scanning)

SunScreen 3.2 integrates features of two earlier SunScreen products—SunScreen EFS and SunScreen SPF-200. It includes two encryption technologies—SKIP (Simpley Key-Management for Internet Protocols) and IPsec/IKE (Internet Protocol Security/Internet Key Exchange).

Each physical interface can operate in either of two modes—routing mode or stealth mode. In routing mode, an interface has its own IP address and can perform routing. In stealth mode, an interface has no IP address and no TCP/IP stack and acts much like a bridge, operating at the MAC (Media Access Control) layer.

SunScreen consists of two primary components: the *screen* (a firewall component), which filters (or screens) packets and provides encryption/decryption, and an *administrative station*. These components can be installed on a single machine or on separate boxes. Typically, a screen and administrative station exist on each location that requires access restriction.

SunScreen can be administered from a GUI or from the command line. You can also administer SunScreen remotely from any Java 1.1.3-compatible browser.

Configuring SunScreen

Configuring SunScreen, like configuring any firewall, consists of turning an organization's security policy into a set of rules that can enforce the specified behavior. When SunScreen is first installed, there is only one policy installed and it's called Initial.

Others must be added to implement security policy as an ordered set. For example, additional rules will allow clear or encrypted communications, based on your criteria.

Rules contain source, destination, and type of service descriptions, along with what action should be taken when packets meet the specified criteria. This is not unlike setting up an email filter, where you might state that any messages containing the word "meeting" in the subject line will be forwarded to your subordinate.

The ordering of rules is important. As soon as there is a match on the selection criteria, the associated actions are carried out and no more rules are applied. If policy does not permit FTP to be used, for example, you can add a rule that looks like this:

```
ftp * * DENY
```

The basic syntax for a rule is:

```
<service>  <source_address>  <destination_address>  <action>
```

The action is always ALLOW or DENY. Optional fields following the fields in the basic syntax include:

Service. The name of an individual network service such as FTP or Telnet.

Source_address. A single network address, range of addresses, or group of addresses from which the packet derives.

Destination_address. A single network address, range of addresses, or group of addresses to which the packet is being sent.

Action. What the "screen" should do with the packet.

Time. An optional field for establishing rules based on time of day.

Screen. An optional field identifying a screen name.

Comment. An optional field that allows someone creating a rule to enter a useful description.

Please refer to http://docs.sun.com for additional documents describing the use and configuration of SunScreen.

SunScreen Lite

SunScreen Lite is a stateful, packet-filtering firewall that uses a subset of the features provided by the SunScreen product. It can:

- Provide basic packet filtering
- Be administered from a remote administrative station
- Be used in virtual private networks (VPNs)
- Display data for SunScreen types and data fields
- Be used for secondary machines in a centralized management group

SunScreen Lite uses SKIP and IKE for encryption. It does not allow physical interfaces to operate in stealth mode and cannot support more than two routing interfaces.

It cannot be a member of a high-availability cluster, does not support proxies, cannot create time objects, and ignores time-of-day fields in rules.

Please visit http://docs.sun.com for additional documentation on SunScreen Lite.

Other Solaris Firewalls

The overall goal for a firewall is to create a virtual network perimeter around your network and enforce rules consistent with organizational policy about who and what gets through. There are a number of firewall products available for Solaris systems other than SunScreen. Those that we know of by experience or reputation include the following:

- Checkpoint Firewall 1
- Raptor
- Gauntlet/TIS Toolkit

DNS and Email Configuration

In Chapter 7, we discussed the configuration and management of DNS. Setting up your name server is an essential step in establishing your Internet presence. You should be sure to set up the obvious servers. If your domain were `foo.com`, you would want to set up `www.foo.com` and probably also `mail.foo.com` and `ftp.foo.com`. Set up a single mail exchanger, but back it up with a secondary one, maybe even a tertiary one, to ensure that your mail services will not be disrupted if your primary mail server has problems. For many organizations, the movement of mail in and out is one of the primary conduits for business, and productivity comes to a halt if this exchange of messages stops. As with hostnames, the use of obvious email addresses for organizational functions will facilitate communications with your customers and users. We suggest that you consider establishing some additional predictable email addresses in addition to the standard ones (e.g., postmaster): info, webmaster, jobs, and sysadmin.

You can distribute mail to groups of individuals by using email aliases if this is appropriate. For example, if mail addressed to *sysadmin@foo.com* should be received by four different individuals, you can set up an alias that causes this to happen.

Having a single mail server will simplify email addressing for everyone who writes to your users and also makes it easier to manage email services for individuals who move from one part of the organization to another. The only real drawback is that the larger your user base, the more likely it will be that there will be username conflicts. Though maintaining unique UIDs and unique usernames across an organization is the ideal, you might have a Jane Doe and a John Doe strongly identifying with the username jdoe. Another useful technique for reducing the identity conflicts in your organization is to provide aliases for such users. If Jane Doe can have mail addressed to *jane doe@foo.com* or *jane.doe@foo.com* as well as *jdoe2@foo.com*, she might be happier handing out her business cards.

sendmail Configuration

They used to call it *sendmail hell*, and anyone with guts enough to get into sendmail's rewriting rules had an instant reputation for being a hacker (in the good sense of the word). It's getting much better. Recent versions of sendmail (e.g., 8.9.3) allow you to configure major features by adding them to an m4 configuration file and then using the m4 tool (e.g., GNU m4) to create your sendmail configuration file, sendmail.cf. The following listing shows a generic solaris2.mc file with one additional line at the bottom.

```
divert(-1)
#
# Copyright (c) 1998 Sendmail, Inc.  All rights reserved.
# Copyright (c) 1983 Eric P. Allman.  All rights reserved.
# Copyright (c) 1988, 1993
#     The Regents of the University of California. All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#
#
#  This is a generic configuration file for SunOS 5.x (a.k.a. Solaris
2.x)
#  It has support for local and SMTP mail only.  If you want to
#  customize it, copy it to a name appropriate for your environment
#  and do the modifications there.
#
divert(0)dnl
VERSIONID('@(#)generic-solaris2.mc    8.8 (Berkeley)
5/19/1998')
OSTYPE(solaris2)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
define('confDOMAIN_NAME,'foo.com')
```

We added the last line to illustrate how you can further tweak the sendmail.cf file you will be creating. The sendmail 8.9.3 distribution includes more than a hundred of these variables. This particular one represents the variable that will be used (if you specify it) to compose the domain. This line would be needed only if your host could not determine this on its own. For those of you who have spent any time inside the sendmail configuration file, this corresponds to the $j macro.

In addition, because sendmail's default behavior is better in terms of security (e.g., no relaying for outsiders by default), you don't have to work as hard to configure it with more conservative options. There are several features you'll probably want to use. You can also set up a database of known spammers and populate it from one of the sites keeping track of the more vigorous spam generators. In addition, you can modify some of your sendmail parameters to lessen the chance that your sendmail service will be used against you. You want to be sure that no one can overwhelm your mail system by sending volumes of large messages, straining the system. By limiting the speed with which sendmail responds as well as its timeout options, you can prevent this from happening. You should browse through the sendmail release notes for the particulars. It's almost as thick as the book *sendmail* by Brian Costales (Costales 1997)—so find a comfortable chair.

sendmail security is significantly improved, it seems, with every release. A significant amount of sendmail security, however, still rests with you. Here are some important pointers:

- Don't leave your aliases, /etc/aliases file, or the NIS map writable.
- Ensure that all files that sendmail reads—its configuration file, lists of domains for which it accepts mail, spammer databases, and so forth—can be altered only by root.
- Make sure that .forward files cannot be created or changed by anyone other than their rightful users.
- Outlaw :include: files in your aliases.

Another email issue that you should be concerned with is user privacy. User mail files (stored by default in /var/mail) should not be readable by anyone other than the owner. Under certain circumstances, anyone reading someone else's email is guilty of a violation of privacy. Be cognizant of organizational policy as well as federal law in managing your users' inboxes.

Client Mail Protocols

sendmail isn't the only software that sysadmins have to deal with in today's networks. Most client non-Unix systems will use POP or IMAP to fetch their email from your mail servers. The typical mail client—such as Eudora or Microsoft Outlook—uses POP or IMAP to retrieve mail and SMTP to send it. Generally, the protocols used are POP version 3 (POP3) and IMAP version 4 (IMAP4).

POP3 is a dead-simple protocol. In fact, it has only a few commands, which you can see in Table 19.3.

Table 19.3 POP3 Commands and Actions

COMMAND	ACTIONS
user	Identifies the user connecting to the service
pass	Announces the password
stat	Makes a status request (tells you how many messages are in your inbox)
list	Lists the messages and lengths
retr	Downloads a message
dele	Deletes a message
quit	Exits

Note the conversation ensuing between a POP3 user and the popper process on the host:

```
telnet spoon 110
Trying 220.203.27.111...
Connected to spoon.
Escape character is '^]'.
+OK QPOP (version 2.53) at spoon starting. <2512.935598886@spoon>
user sandra
+OK Password required for sandra.
pass NotMyPswd
+OK sandra has 2282 messages (32289278 octets).
stat
+OK 2282 32289278
dele 1
+OK Message 1 has been deleted.
quit
+OK Pop server at spoon signing off.
Connection closed by foreign host.
```

To set up POP3, you need the popper executable plus the line in your `/etc/inetd.conf file/`, as shown in the following example. Users can retrieve their mail from almost anywhere. The most complicated part of running POP is keeping your users in touch with where their email is at any point in time. Unless they are careful to select the *leave mail on the server* option when they are popping in, their mail will be gone when they come back to the office. Make sure that they understand this. Once some of their mail is on their home systems, some is on their laptops, and some is at work, they will begin to feel more fragmented and frustrated than usual.

```
# for the popper daemon:
pop3    stream  tcp     nowait  root     /usr/sbin/popper    popper -s
# for IMAP clients:
imap    stream  tcp     nowait  root     /usr/local/etc/imapd    imapd
```

IMAP4 is another mail service, but one that supports email folders. Many PC mail clients now use IMAP4. Again, be careful to inform your users about the implications of downloading mail from a particular location.

One problem that you need to watch out for with POP and IMAP clients is a mail buildup on the server. Many mail clients keep a map of which messages have been deleted by the user but do not actually delete the messages on the server. From the users' point of view, they are managing their mail holdings responsibly. Meanwhile, their mail on the server may be larger than 100 MB per user, dominating the space available for incoming mail.

Configuring FTP Services

In general, there is not much that needs to be done to configure the FTP service on a system. In almost every case, this service will be up and running when a system is first installed and booted. If your users' accounts are configured to make use of non-native shells such as bash and tcsh, you will need to add lines to your `/etc/shells` file before your FTP service will allow these users to log in. In this case, add the full pathname to the respective shell (e.g., `/opt/gnu/bin/bash`) to this file.

Anonymous FTP services must be explicitly configured if you require this service. We advise against using Solaris FTP for your Internet services—especially if you are supporting anonymous access. The Solaris FTP is a perfectly functional, but limited, FTP service. For your Internet services, we suggest that you download and install `wuarchive-ftpd` (commonly referred to as `wu-ftpd`)—a replacement FTP daemon for Unix systems, and the most popular FTP daemon on the Internet. With considerably better security, logging, and differentiation between users and virtual users, `wu-ftpd` is the choice of responsible sysadmins. The following URL contains lots of useful information and sample scripts: www.landfield.com/wu-ftpd.

Abuses of FTP include using it as a repository for pirated software and other copyrighted material and pornography. In addition, misconfigured FTP setups have sometimes allowed processes to be run under the FTP user ID.

To set up anonymous FTP service, you should determine the purpose of the service—whether it is intended to serve as a drop-off only, as an archive only, or must support both functions. If your FTP service is an archive, restrict access so that it is read-only to the FTP user. If your FTP service is for drop-offs only, don't provide read access.

FTP sites can also have an impact on the performance of the system on which they are running. In a prior position, one of the authors noted that a Sun client had slowed to a point at which it was almost useless. As it turned out, an undergrad assistant in the department had mirrored a collection of image files (including "Todd in Tights"). The explosion of logins dragged the system to its knees.

To avoid abuses of your FTP site, scan its contents regularly, looking for suspicious files. Don't forget to use the *-a* option so that you will spot any hidden files or directories. We also strongly suggest that you read the following guidelines produced by CERT: ftp://ftp.cert.org/pub/tech tips/anonymous ftp config and ftp://ftp.cert.org/pub/tech_tips/anonymous_ftp_abuses.

NOTE Make sure that the `wu-ftpd` package you install is the correct one for your Solaris version. It may not compile or work properly otherwise.

Managing Your Web Presence

When one of the authors lived in California, there was a very conspicuous billboard on Route 101, south of San Francisco. It said something like "My grandma can make a better Web site than your grandma." If you're putting up a Web site for your company, you've got a lot of grandmas competing with you for browsers' eyeballs. You've also got a lot of other companies, a lot of independent contractors, a lot of college students, a lot of hobbyists, and a lot of kids. You'll want your site to be not only attractive, but also effective and responsive. But that's only the *fun* part of building a Web site. You'll also want to keep it up as close to 24 x 7 as possible. Configuring a Web site involves a series of steps. There are a number of important configuration files to set up once your Web server software is installed—and some new log files to maintain, process, and clean up after.

Solaris 9 includes a binary distribution of Apache that can be quickly configured to provide Web services on this optimized-for-Web operating system. The iPlanet Web server is also available for Solaris and is often preferred for performance-hungry sites. In this section, we look briefly at each of these products and provide a brief tutorial to help you set up a Web site quickly. We will also explain some of the major differences between these two Web servers, most of which revolve around how the servers are configured rather than how they operate.

The most popular Web server at this point in time is Apache—and for good reason. Its configuration files are easy to understand and modify. It performs well and has all the features most of us could want. Apache is also both free and open source. Though Apache may be surpassed in performance by iPlanet, it is still used on many large Web sites and sites with large numbers of virtual servers. To keep us all on our toes, Sun is now moving over to the name “Sun ONE Web Server.” In this chapter, we will refer to it simply as iPlanet.

Regardless of what Web server software you choose, Web servers have a lot in common. Whether you’re working with Apache, iPlanet, or any other Web server software, you can expect to find some way to configure:

- The location of Web pages
- The default documents (the pages which are displayed when only directory names are included in URLs)
- The location and content of log files
- The port that the server will run on (almost always port 80) and the user ID under which the process will run
- How visitors will be authenticated

Specifying Ports and Protocols

As indicated earlier, the well-known (or *reserved*) port for `httpd`, the HTTP daemon, is port 80. Whenever you visit a Web site without specifying otherwise, your browser is communicating with this port. To specify an alternate port, you include it at the end of the home page URL, like this: http://www.somesite.com:8888.

You are seldom likely to use this syntax, but might run into it if the site is run by an ordinary (not a system) user or if the site is *cloaked* (i.e., hidden by virtue of running on an unexpected port).

The HTTP protocol, as you probably well know, includes directives: `get`, `put`, `post`, `link`, `unlink`, and `delete`. Specifications for the HTTP protocol also identify the form of a URL as http:// followed by the hostname (e.g., *www.foo.com*), an optional colon and port number (e.g., :8888) and an optional path (needed if you are accessing other than the home page). If you wanted to go directly to a subsidiary page on a cloaked site, you might use an address like this: http://www.foo.com:8888/contributor/notes.

If your site uses SSL in part or in whole, some of your URLs will look like this: https://www.foo.com/login.jsp.

The `https` preface indicates that the connection is encrypted using SSL. SSL connections normally run on port 443. The port does not have to be included in the URL as it is implied when the `https` protocol is specified. Besides, most of the time, visitors will end up using https because they are redirected to this port, not because they have typed the **https** themselves.

Support for Virtual Web Sites

Both Apache and iPlanet provide the means to set up numerous virtual sites on a single server. In order to run more than one Web site on a server, you need to vary one of the following pieces of information:

- The IP address that is connected to the site (i.e., you can set up virtual IP addresses so that the system responds to more than one address)
- The port that is used to communicate with the site (e.g., you can run one Web site on port 80 and another on 8080)
- The name by which the server is known (e.g., a site may be known both as www.foo.com and www.toadsnfrogs.org)

As long as one of these pieces of data is different, you can operate more than one Web site. The third option, defining each Web site by the name that is used to connect to it, is the preferred method because it doesn't require additional IP addresses, nor does it rely on the use of odd ports that would need to be included in the URLs.

When you differentiate sites by server name, you are making use of a feature known as Host Headers that was introduced into the HTTP protocol in version 1.1. This feature allows the name included in the client browser to be carried in the header of the client's request. This allows the server to identify the site requested. The client is unaware that it is communicating with one in a set of sites.

NOTE Very old browsers that do not support the Host Header feature will not be able to reach sites which are defined this way.

In the following sections, we will take a look at how virtual Web sites are configured in each of these servers.

Apache

Apache is an extremely popular and versatile Web server. It acquired the name "Apache" because it was a homonym for "a patchy" server, meaning that it was extensible, not that it requires lots of patches to work. Apache has been around since April 1995 when the first public release became available.

Apache is not only free, it's open source. This means that developers can obtain and modify the code for their use. The version included in Solaris 9 is 1.3.20.

In early versions of Apache, several configuration files were used to contain different types of configuration information. In recent and current versions of Apache, however, only a single configuration files is used. This single configuration file is called `httpd.conf` and is a text file that contains two things: comments (lots of them) and *directives* (commands that tell the server how to behave). As with the bulk of Unix configuration files, Apache's comments begin with the "#" character. Directives can be single or multi-line.

Apache is extensible through the addition of modules that can be included in the `httpd` binary at compilation time and by the configuration of many features that can be turned on or off in at any time in the configuration file.

Minimal Apache

With the barest installation of Apache, you could get a basic Web site up simply by dropping your prepared Web pages into the default document directory and starting the server. Let us assume that you have a tarball containing the text and image files that comprise a Web site.

The first thing you would want to do is copy and modify the example configuration file and establish the location of your document directory (the *DocumentRoot* in Apache lingo). On an established Apache site, you can find this information by looking for the DocumentRoot directive with a command such as this:

```
# grep DocumentRoot /etc/apache/conf/httpd.conf
DocumentRoot /opt/apache/htdocs
```

The DocumentRoot tells you where to untar your Web files from your tarball.

```
# cd /opt/apache/htdocs
# tar xvf /tmp/website.tar
<tar output>
```

If you then start up the Apache server with the following command, you will have a Web site up and running. Though this simple setup might not be all that you need, the default Apache configuration runs well and provides good service for the typical Web site.

```
# /usr/apache/bin/apachectl start
```

To ensure that this site starts up whenever the server is booted, copy the `apachectl` script into the `/etc/init.d` directory and make the following links if they do not exist:

```
# ln /etc/init.d/apachectl /etc/rc2.d/S98apachectl
# ln /etc/init.d/apachectl /etc/rc1.d/K56apachectl
```

Your default Apache configuration will now start whenever the system boots, will initiate a number of server processes whenever Apache is started, and will provide a larger number of processes (make sure this isn't the multi-threaded server) to meet increased demand. Your log files will be stored in /opt/apache/logs and will be called access and errors.

Starting and Stopping Apache

The Apache HTTP daemon, httpd, is started up through the init process. A typical start/stop script would look that shown in the following listing.

```
#!/bin/sh
#
# Start WWW httpd for www.foo.com.
#
httpd="/usr/sbin/httpd-apache"
httpdconf="/etc/httpd-apache.conf"
startbanner="Starting Apache WWW daemon for www.foo.com."
stopbanner="Stopping Apache WWW daemon for www.foo.com."
pidfile="/etc/httpd-apache.pid"
case "$1" in
'start')
     if [ -f $httpdconf -a -f $httpd ] ; then
          echo "$startbanner"
          $httpd -f $httpdconf &
     else
          echo "$0: unable to find $httpdconf or $httpd"
          exit
     fi
     ;;
'stop')
     if [ -f $pidfile ] ; then
          echo "$stopbanner"
          PID='cat $pidfile'
          /usr/bin/kill ${PID} 1> /dev/null 2>&1
     else
          PID='/bin/ps -ef | grep httpd-apache | grep -v grep | awk
'{print $2}''
     if [ ! -z "$PID" ] ; then
               echo "$stopbanner"
               /usr/bin/kill ${PID} 1> /dev/null 2>&1
          fi
     fi
     ;;
*)
     echo "Usage: $0 { start | stop }"
     ;;
esac
exit 0
```

If stored as /etc/init.d/httpd and hard-linked as /etc/rc3.d/S98httpd and as /etc/rc2.d/K55httpd, the daemon would be stopped in run state 2 and started in run state 3. If you needed to restart the daemons manually, you might do this:

```
cd /etc/init.d
./httpd-apache stop;./httpd-apache start
```

NOTE We said *daemons* because there will likely be many more than one running. The Apache server will fork additional processes as needed to help support the load.

Configuration Files

Apache configuration files are fairly straightforward and easy to understand. Of the original four configuration files, only the `httpd.conf` file is used today. The other three files generally appear in the configuration directory for the sake of continuity, but are relegated to being nothing but placeholders.

If you were called to modify the configuration on an unfamiliar system, you could generally depend on being able to locate the configuration files by first looking at the startup script in `/etc/init.d`. As previously shown, this file includes the location of the configuration files and the daemon process.

Directives are commands that control how Apache executes. Many are single-line commands such as these:

```
DocumentRoot /opt/web
Port 80
ServerName www.foo.com
```

These single-line directives simply assign values to the variables specified. Other directives encompass a number of lines and include both a command that starts and a command that ends the directive. Notice that these start and stop lines have a resemblance to HTML syntax.

```
<Directory /usr/local/httpd/htdocs>
Options Indexes FollowSymLinks
</Directory>
```

This set of lines define how accesses to the directory specified will be treated. The options specify that directory browsing is enabled (i.e., a directory listing will be presented to the visitor if none of the default documents exist in the directory specified) and that symbolic links will be followed if they exist. We will look at the security and performance implications of these choices later in this chapter.

If the default Apache configuration does not suit you, don't despair. There are several hundred directives that you can include in Apache's configuration files. We will describe only a handful in this chapter. You can obtain a full list, including considerable detail about their use, by visiting the following Web site, which provides information on each of the available directives: httpd://www.apache.org/docs/mod/directives.html.

The directives in the default configuration file will include lines such as these:

```
BindAddress foo
ErrorLog logs/www.foo.com-errors
Group staff
HostnameLookups on
KeepAlive 5
KeepAliveTimeout 15
MaxClients 150
MaxRequestsPerChild 30
MaxSpareServers 10
MinSpareServers 5
PidFile /etc/httpd-apache-foo.pid
Port 80
ScoreBoardFile logs/www.foo.com-status
ServerAdmin webmaster@foo.com
ServerName www.foo.com
ServerRoot /var/WWW/apache/foo
ServerType standalone
StartServers 5
Timeout 400
TransferLog logs/www.foo.com-access
User nobody
```

As you can see from this list, the base of the Web directory (`/var/WWW/apache/foo` in this case) is specified as the *ServerRoot.* The configuration (`conf`) and `log` directories are within this directory. The `conf` directory contains the remainder of the configuration files.

Directory Structure

The directory structure that your site will use depends, of course, on your preferences and the setup on the configuration files listed previously. However, the form most Web sites take looks roughly like this:

```
               +--- conf
               |
ServerRoot ----+--- logs

               +--- cgi-bin
               |
DocumentRoot --+--- icons
               |
               +--- images
               |
               +--- [content subdirectories]
               |
               index.html (or other default document) and other pages
```

By default, your visitors will have access only to files within the DocumentRoot subdirectory and below. They will not have access to your configuration and log files. Directories included within your directory structure should have permissions set to 755 (rwxr-xr-x), html files to 644 (rw-r-r--), and cgi scripts to 755 (rwxr-xr-x) for the Web site to work as intended. If you have multiple people supporting your Web site, you may open up group permissions, but continue to restrict general access.

Busy Web sites will generate lengthy log files. You should be careful to monitor space usage and to compress and archive older files as needed. To collect this information and not use it, on the other hand, is inadvisable. Log files are a source of considerable information on the popularity and use of your site. Because none of us has the time or interest to read megabytes of get requests each and every day, there are tools that can help you digest your log files. Of the free tools, the best we have found is *Analog*. It runs through even very large log files quickly and generates an HTML report, including graphs detailing file accesses and displaying your hits over the span of a month or daily averages, depending how what you specify in your configuration file. We found it extremely easy to set up and use, and the reports it created provided some insights we didn't expect. You can learn more about it or obtain Analog from http://www.analog.cx/.

If you can afford the resources, it's a good idea to maintain both an internal development Web site and a public Web site. This permits you to test changes to your Web site and then *push* (i.e., mirror) your Web site when the changes have been verified; duplicate your site if it is appropriate to do so; or quickly overwrite your site if, in spite of your efforts, someone breaks in and replaces your corporate image with pictures of Todd (see *Configuring FTP Services*). To this end, we recommend the Mirror script (`mirror.pl`). Mirror is a package written in Perl that uses FTP to duplicate an entire directory hierarchy on another system. It is not necessarily restricted to use as a push resource for Web sites, but it serves this purpose very well. Simple configuration files identify the source and destination hosts, user, and password (be careful to protect these files), along with expressions that detail what should be excluded from the push and so forth. You can learn about Mirror and obtain the package from http://sunsite.doc.ic.ac.uk/public/packages/mirror/mirror.html.

Tuning and Configuring Apache

There are a number of things that you can do to increase the performance of your Apache server although the default Apache configuration works well for the average site. If you are trying to eke out the best possible performance, you might decide to recompile Apache from source files instead of going with the binary distribution included with Solaris. This would allow you to fine tune your configuration by omitting and modules that you do not need for your site. For example, if you have no need now or ever to support personal Web sites (i.e., those that allow individual users on a server to publish Web pages from their `public_html` directories), you could omit this feature from your compiled code, and your `httpd` will, as a result, be somewhat smaller.

Runtime Tuning

Most performance tuning for Apache servers involves one of two things. The first involves changes to the directives that determine how many Apache processes will start up when Apache is started and how number swells and shrinks with demand. These directives provide your Apache server with the ability to quickly respond to client requests, while not running so many processes that resources are wasted. The second involves the way that directories are configured. There are two options to pay close attention to.

One of these options is FollowSymLinks. This directory option specifies whether or not symbolic links with the Web content directories will be followed to resolve a page request. Though this might at first appear to be a setting that would consume resources, the opposite is true. When FollowSymLinks is turned off (i.e., `-FollowSymLinks`), every element in a URL must be examined to determine whether or not it is a symbolic link. This adds to the file system accesses needed to retrieve every file on the site or the directory in which this setting is turned off. Similarly, `SymLinksIfOwnerMatch` directives should be avoided. Both of these directives will issue system calls to read and evaluate files to determine whether they are symbolic links and who owns them.

`HostnameLookups` should be off unless you need to perform DNS lookups on clients. Similarly, Allow from and Deny from directives should be avoided if you don't need them. They, too, will cause additional DNS lookups to verify whether the hosts are among those allowed or denied.

If you allow `.htaccess` files to be used to modify the controls on directories outside of `httpd.conf`, it is wise to restrict their use to only those directories where this control is needed. If you allow these files anywhere within your Web page directory structure—for example, with lines such as these:

```
DocumentRoot /opt/apache/htdocs
<Directory />
    AllowOverride all
</Directory>
```

The server will look for `.htaccess` files in `/`, `/opt`, `/opt/apache`, and `/opt/apache/htdocs`.

The directives that control the number of Apache processes are StartServers, MinSpareServers, and MaxSpareServers. These determine, respectively, the number of server processes that start when Apache is first started, the number of idle processes that must exist at all times when Apache is running, and the maximum number of idle servers that are allowed to run when Apache is running. The desired effect is that you have enough idle servers to respond quickly to bursts of activity, but not so many that you're consuming resources that might be needed by other applications.

These and other Apache tuning considerations are addressed in this document:

```
http://httpd.apache.org/docs/misc/perf-tuning.html
```

NOTE **The directives that control the number of Apache processes that run at any point in time will be moot or will take some different form in a multithreaded server.**

Disallowing Directory Browsing

Directory browsing is the term used to describe what happens when a visitor enters a URL and, instead of displaying a default page, a directory listing is returned instead. This does not happen if any of the documents specified as defaults for the Web site or for that particular directory exists in that directory. In other words, directory browsing describes what happens when no default document exists in the specified directory.

For most sites, this is not the behavior that is most desirable. Unless the files being listed are intentionally being offered for downloading, giving a directory listing makes the site appear amateurish and exposes your directory structure. There are two ways to ensure that this doesn't happen. One is to be sure that every directory contains a file whose name is in the DirectoryIndex list. The other is to turn off directory browsing for the entire site or for the individual directory. This is done by changing the Indexes option to `-Indexes`, as shown in this example:

```
# At the server level, switch off directory browsing.
<Directory /opt/apache/htdocs>
Options -Indexes FollowSymLinks
</Directory>
```

If you want to turn directory browsing on for a particular subdirectory, you can reverse this setting in the directory directive that applies to that particular directory. For example:

```
# Allow directory browsing for download directory.
<Directory /opt/apache/htdocs/downloads>
  Options +Indexes
</Directory>
```

Note that almost any directive that you can use within the `httpd.conf` file can be used within a directory directive (sometimes referred to as a "directory container"). This allows you to have very different settings for some Web directories and provides the basis for virtual Web sites.

Adding Virtual Web Sites

Name-based virtual hosting in Apache is easy to configure and support. Before editing the `httpd.conf` file, you need to be sure that each of the additional names to be associated with the server have been set up as CNAMEs (aliases for the server) in your DNS server. Once this is done, you can add a set of lines like those shown below for each virtual server that you will be supporting.

```
NameVirtualHost *

<VirtualHost *>
ServerName www.foo.com
DocumentRoot /opt/apache/htdocs
</VirtualHost>
```

```
<VirtualHost *>
ServerName www.frogsntoads.org
DocumentRoot /opt/apache/frogsntoads
</VirtualHost>
```

When the additional directory has been populated and Apache restarted, the server will respond to two names with Web pages from the proper directory. The server name will be included in the host header information provided by the client.

TomCat and Perl 5

Included in Solaris 9 along with Apache are two invaluable tools: Perl 5, the primary scripting language used for Common Gateway Interface (CGI) programming, and TomCat, the public domain tool for working with JavaServer Pages (JSPs).

Perl was introduced in Chapter 12 and is the primary scripting language for Common Gateway Interface (CGI) programming. Perl scripts are used for creating Web pages, analyzing traffic, processing online forms, and managing Web sites in other ways.

Tomcat is referred to as "the official reference implementation of the Java Servlet 2.2 and JavaServer Pages 1.1 technologies." What this means is that it provides the type of services that many would refer to as a Java compiler. It provides a way for your Web site to run servlets—memory-resident Java programs. These servlets run inside a servlet *container*, in other words, Tomcat. Because servlets are memory resident, they are much faster. They avoid the overhead of process creation and subsequent cleanup.

Tomcat is an open source tool that is much like Allaire's JRun.

iPlanet Web Server

iPlanet is a Web server which can be configured through its configuration files (see Table 19.4) or through the admin server—an administrative interface available on the Web that typically runs on port 8888. The primary configuration files for iPlanet Web servers are `magnus.conf` and `obj.conf`. They typically reside in a directory such as `/opt/iplanet/servers/config`, along with a number of other files. What goes into each of these files depends on the version of iPlanet that you are running.

The `obj.conf` and `magnus.conf` files contain much of the same type of information that is in Apache `httpd.conf`. This involves a series of instructions (called directives just as they are with Apache) that tell the iPlanet Web Server how to process client requests. These directives are all written in a special language called the Netscape Server Application Programming Interface (NSAPI). The iPlanet Web Server comes with a set of predefined server application functions, but the server can be extended if you choose to create additional instructions in this language. Most users make minimal changes to the configuration of their iPlanet servers.

Table 19.4 iPlanet Files

FILE	PURPOSE
`magnus.conf`	Contains global server initialization information
`server.xml`	Contains initialization information for virtual servers and listen sockets
`obj.conf`	Contains instructions for handling requests from clients
`mime.types`	Contains information for determining the content type of requested resources

Some of the more popular directives available in iPlanet include:

AuthTrans (authorization translation). Verifies authorization information (such as name and password) sent in the request.

NameTrans (name translation). Translates the logical URI into a local file system path (used for redirects).

PathCheck (path checking). Checks the local file system path for validity and check that the requestor has access privileges to the requested resource on the file system.

ObjectType (object typing). Determines the MIME-type (Multipurpose Internet Mail Encoding) of the requested resource (for example: `text/html`, `image/gif`, and so on).

Service (generate the response). Generates and returns the response to the client.

AddLog (adding log entries). Adds entries to log file(s).

Error (service). Executes only if an error occurs in one of the previous steps.

Copies of the NSAPI Programmer's Guide are available online at a number of locations. We recommend that you familiarize yourself with this document if you manage an iPlanet Web site. One source is http://enterprise.netscape.com/docs/enterprise/60/nsapi/contents.htm.

Summary

Running an Internet site is a major responsibility. While you are providing your users with access to Internet resources, you are also providing the outside world with limited access to your organization's resources, and therefore influencing its public image.

A firewall is strongly recommended, whether you spend the money to buy a commercial tool or set one up yourself with a spare system—but do the latter only if you have the time and resources to do it right. How successful you will be at limiting what comes into and goes out of your Internet site will depend on how well your organization's policies are specified and implemented and the degree to which your organization backs your efforts to limit access.

Regardless of whether you are able to sit behind a firewall, you cannot take security for granted. Some other helpful security reminders include the following:

- Your services are secure only if they are properly configured. Take advantage of organizations like CERT that report on security problems and fixes.
- It is important to use the latest versions of services, such as `sendmail`, that tend to be vulnerable to attack. The latest software will include the fixes for security holes discovered in earlier versions and may provide you with some useful configuration options you didn't have with the older release.
- Never assume that, just because there are dangers *out there,* there are no dangers from the inside. Be conscious of security in general, not just as it pertains to the Internet.
- Don't allow your Internet security setup to lull you into a false sense of security. Know how to monitor your systems and watch for signs of trouble. Consider scanning your systems from the outside, but only *with* official permission to do so!

CHAPTER

20

Coexisting with other Empires

One of the authors once worked in an office where everyone's desk, from the office chief to the lowliest of workers, held a Sun workstation. There was bandwidth to boot and fiber just under the floor awaiting a planned move to the next generation of networking. Because all of the systems were basically the same, few tools were needed to manage the network and configuration changes could be propagated throughout the network with little effort.

Most offices today bear little resemblance to this ideal of roughly 12 years ago. Most of today's networks consist of a mix of systems—Sun servers, PCs running Solaris, PCs running Windows, PCs running Linux, and Macintosh systems—with all of these systems running different programs for users with vastly different sets of needs and skills. It's hard to define a set of tools to manage this diverse collection of systems with the same ease that we managed homogeneous networks of Sun systems. Yet, this is the environment that we work in today. This is the environment that we expect you work in, as well.

Of the non-Unix systems that we have to deal with, an increasing number are running some version of Microsoft Windows. The ability to share resources and interoperate with other types of systems is *not* one of Windows' (even Windows 2000's) strengths; in fact, this is probably its greatest weakness. Microsoft Windows is what some of us like to call an "80 percent operating system." Were we left with only what we can get from Microsoft, our fate as system administrators in mixed Unix/Windows environments would be unbearably difficult. Thanks to scores of third-party developers, we have many technologies to help bridge the gap between these two diverse technological camps. We also can't overlook the efforts of Sun Microsystems and other devotees of open systems in making it possible for such technologies to exist.

Years ago, collections of Windows or Macintosh systems seemed to be grouped together somewhere far from the Unix systems that were ours to manage. Today, they're sitting side by side with Unix systems, cabled into the same network equipment, and using the same files. In this chapter, we hope to provide some guidance on how you can manage a mixed architecture environment like this while averting chaos.

At the same time that coexisting with Windows remains a significant challenge, often answered by separate staffs that cross-pollinate only when absolutely essential, co-existing with other Unix systems has become easier.

Solaris Meets GNOME

In the first edition of this book, we placed a lot of emphasis on coexistence with Windows systems (i.e., The Evil Empire), but recently Sun has taken a significant turn toward compatibility with an empire that is anything but evil—the world of Linux and of open source computing. The most obvious outcome of this friendliness between the Unixes has been the inclusion of the GNOME 2.0 beta desktop in Solaris. The inclusion of GNOME in Solaris represents far more than a token gesture of good will: all future development of the Solaris desktop will be in GNOME, not CDE. This is a major direction for Sun and an exciting development for Unix users.

Not only is GNOME a friendly desktop environment, it provides numerous desktop tools and applications that are likely to make it an instant hit with users who are not already familiar with it by virtue of being Linux users on other systems or at home.

Adding the GNOME Desktop to Solaris

GNOME stands for *GNU Network Object Model Environment* and is itself entirely built on open source software and open standards. It provides

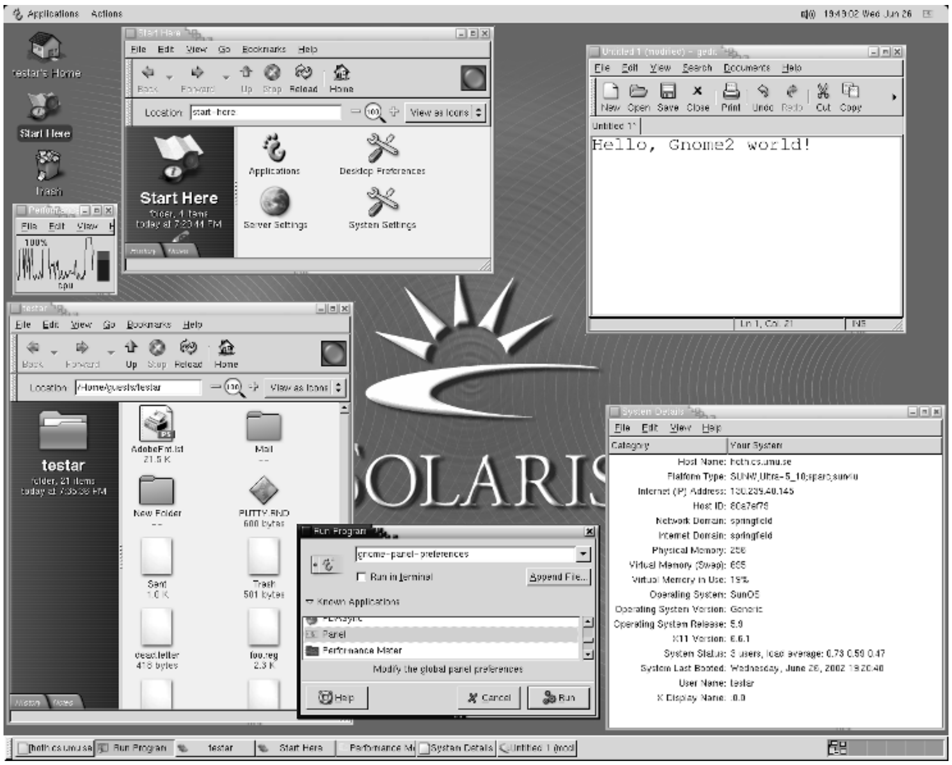
- An intuitive interface
- Numerous user applications
- Web browsers
- And much more

In Solaris 9, GNOME is included on a separate CD that arrives with the OE, entitled "Exploring the GNOME Desktop." Once installed with the installer tool provided on this CD, GNOME is available as an alternate desktop.

GNOME is especially ideal for Unix shops that use both Linux and Solaris systems to alleviate the stress that can be caused by users moving between the two desktop systems on a frequent basis.

The office productivity applications include Web browsers, a word processor, presentation software, a spreadsheet, a graphics editor, an email client, a calendar, a contact manager, a Web publisher, and an image editing tool.

GNOME is a great desktop both for users and developers. It represents a desktop that is simpler for staff who are not especially adept at Unix, especially on the command line. At the same time, it also provides a toolkit (GTK+) for developers, including a built-in framework for building highly accessible desktops for users with disabilities.



```
Hello, Gnome2 world!
```

```
gnome-panel-preferences
Modify the global panel preferences
```

Category	Your System
Host Name:	hoth.cs.umu.se
Platform Type:	SUNW,Ultra-5_10;sparc;sun4u
Internet (IP) Address:	130.239.40.145
Host ID:	80a7ef79
Network Domain:	springfield
Internet Domain:	springfield
Physical Memory:	256
Virtual Memory (Swap):	655
Virtual Memory in Use:	19%
Operating System:	SunOS
Operating System Version:	Generic
Operating System Release:	5.9
X11 Version:	6.6.1
System Status:	3 users, load average: 0.73 0.59 0.47
System Last Booted:	Wednesday, June 26, 2002 19:20:40
User Name:	testar
X Display Name:	:0.0

Figure 20.1 GNOME on Solaris.

Though GNOME is a big change in direction for Sun, it is not a sudden one. Sun has been involved in GNOME development since it joined the GNOME foundation in the year 2000 and is playing an active role in GNOME's continued development. This means that GNOME will continue to be well integrated into Solaris.

NOTE **Most existing CDE tools should run under GNOME as well.**

GNOME is developed by the same group as GNU/Linux with source code ready and available for further development by a worldwide community. Figure 20.1 displays a typical GNOME desktop on Solaris.

Applications and GNOME

The variety of applications available in the GNOME desktop is fairly extensive. These include:

StarOffice Writer. A tool for creating documents and reports, incorporating text and images. It also allows content to be saved in HTML format.

StarOffice Calc. A spread sheet application, which includes charting.

StarOffice Impress. A presentation tool, including clip art and animation.

Evolution Mailer. An email client, which includes support for audio and video.

Evolution Calendar. A calendar tool for tracking appointments, and remembering company holidays, task deadlines, and meetings. It also provides tools for synchronizing with palm-held devices.

Evolution Contact Manager. A tool for managing contacts, including phone numbers, addresses, and so on. It can also be synchronized with palm-held organizers.

Nautilus. The tool for remote and local file management, complete with thumbnail views of image files and integrated file searching.

Netscape. Of course, a popular Web browser.

Media Player. A tool for listening to music.

Text Editor. A tool that provides simplified editing of text and html files.

Terminal. A tool for connecting to other systems.

Screen Shooter. An image capture tool.

GNOME Help. GNOME Help is also available on the desktop with balloon help features.

GNOME Customization

Like CDE, GNOME provides multiple desktops, which can be selected through a panel at the bottom of the screen. It also provides the user with a choice of desktop themes, control over screen layout, and choice of backgrounds and icons. The customization and tool-rich nature of the interface allow users to quickly become productive and comfortable with the GNOME environment.

The control panel at the bottom of the screen is also easy to use and is likely to seem natural to users who have been working in CDE. Many users may be already familiar with the GNOME desktop from other systems that they use or have used. The same desktop is available on Linux, HP-UX, BSD, FreeBSD, NetBSD, OpenBSD, and IRIX systems.

GNOME Development Tools

Developer tools that come with GNOME include:

GTK+. A C-based, object-oriented user interface toolkit.

ORBit. A small and fast CORBA object request broker (ORB).

Bonobo. The GNOME component architecture library.

GConf. A GNOME network-aware and component-aware configuration management tool.

glib. A common data structure manipulation library.

gdk-pixbuf. An image library for manipulating images in many formats.

libgnomeui. A GUI-related GNOME code library.

libgnome. A non-GUI GNOME code library.

libart. A graphics rendering routine library.

gnome-print. A printing routine library.

libxml. A library for parsing XML data files.

gnome-vfs. A virtual file system library.

Glade. A user interface builder for GTK+ applications.

Accessibility Framework. The ATK API and associated library for accessibility applications and the AT SPI for interfacing accessibility technologies (e.g., voice recognition) on multiple Unix platforms.

Transitioning to GNOME

Sun's adoption of GNOME as an alternative desktop for Solaris users is an exciting development. GNOME is an extremely well designed and popular desktop and, while not abandoning CDE quite yet, Sun is providing the GNOME desktop to give users a desktop that is nearly the ultimate in user-friendliness. This move also meets the needs of organizations that use a mix of Solaris and Linux systems.

The transition from CDE can be taken slowly. CDE will remain available as a login option for an indefinite time. The same functionality will be available in both desktop environments, and the GNOME desktop is especially appealing and intuitive.

Working with Windows

The most common network configuration that the authors have encountered in the last few years involves using Sun servers in server rooms, while Windows clients provide all access to the network for staff. Even those of us tasked with managing Solaris servers exclusively have often done so from the keyboard of a Windows-based PC. Whether you manage both the Solaris systems and Windows system or are merely responsible for keeping the services of the Solaris systems available to your users, some knowledge of Windows will help you understand the challenges and possibilities of marrying these technologies. In this section, we provide an introduction to Windows essentials from the perspective of long-time Solaris system administrators.

Windows Essentials

If you're not already steeped in the proper terminology to use when describing Windows systems, a little vocabulary is in order. Windows can be purchased in *both* server and non-server (i.e., workstation) versions. Server versions provide Internet services such as the IIS (Internet Information Server) Web server, an FTP server, a mail server, and a DNS server, while clients are expected to be used for client software such as browsers, mail clients, and so on.

Some versions of Windows, such as Windows 2000 Advanced Server, can provide additional services and can act as a *domain controller*—similar to a NIS/NIS+ server. From a security point of view, the important distinction is between workstations (Windows 2000 Professional, Windows XP Professional, or Windows 2000 Advanced Server performing as a workstation) and domain controllers.

Another important difference is the one between *local accounts* and *domain accounts.* Local accounts are those that are used only on the local system. They don't exist as far as the domain is concerned. Most Windows 2000 accounts are, however, domain accounts. These are stored on domain controllers responsible for maintaining them and authenticating requests for service (e.g., logins). Although domain controllers can provide file serving, they are not necessarily used for this; their primary role is maintaining account information and authentication.

Users and groups are not implemented identically in Windows 2000 and Unix systems though the concepts exist in both. Both support groups and access control lists (ACLs). Every user in either system is a member of at least one group and can be a member of more. The mechanisms are different, of course.

The Administrator account on a Windows 2000 system is not precisely the same as a root account on Solaris. The Administrator account cannot be served via the domain manager; it is *always* a local account.

TIP The root account can be set within NIS or NIS+, but we strongly recommend that you maintain a root login in your `/etc` files. That way, if ever your NIS/NIS+ service isn't working, you can still log in and fix things.

If you don't remotely administer your Administrator accounts, you can still set the passwords remotely by logging in and using the following command:

```
net user Administrator *.
```

This command will prompt you for a new password.

Windows 2000 does not have the concept of a *nobody* (i.e., superuser on a remote host) user and comes with a guest account that works very differently from the guest accounts that came enabled in much older versions of SunOS. On Windows 2000 servers, the guest account is assigned when the server can find no match among its valid accounts. This is markedly different from the guest account in old SunOS systems, which simply lacked a password (or maybe the password was set to *guest* or *welcome*). Both of these accounts are considered very poor practice today. The guest account should be disabled immediately following installation.

Another difference between Windows and Solaris is that Windows does not enforce the concept of a primary group. In Unix systems, the group assigned to a user in the `/etc/passwd` file is the user's default or primary group. On Windows systems, all user group memberships are set up within the User Manager for Domains tool. Another difference is that Windows systems have an *everybody* group—a supergroup composed of members of any group and, therefore, all users. Unix systems do not have a group that contains all users. The closest match is the *world* group, which, in general terms, refers to anybody *except* the owner and the group associated with the file. Solaris ACLs, now extended to provide secondary owners as well as specifications like "everybody *but* Mallory," are not available in Windows 2000. At the same time, Solaris does not have an equivalent to Windows 2000's *take ownership* attribute; by default, only the owner of a file in Solaris can give it away (i.e., make someone else the owner).

Domain controllers are subdivided into *primary domain controllers* (PDCs), of which you must have one, and *backup domain controllers* (BDCs), which act like slave NIS (or replica NIS+) servers. Changes are made on PDCs and propagated to BDCs in the same manner, and changes are made on NIS servers and propagated to slaves. Each workstation in a Windows 2000 domain is a member of only one domain.

Domain controllers can establish *trust relationships* between themselves, which work in a single direction only. This allows them to share account information from one domain to the next.

Associated with each Windows 2000 account are the same kind of access controls you might expect, coming from a Unix background:

- Groups (so that you can manage service access by group rather than by individual user)
- Password expiration
- Password parameters (e.g., minimum length and whether a user can reuse a password)
- Whether a user can change his or her own password
- The hours during which a user can log in
- Account locking if a wrong password is used a specified number of times

You can use a tool available in Windows 2000 to lock administrator access for anything other than local logins—the antithesis of what we're trying to do in this chapter (facilitate remote administration), but similar to the feature in Solaris (see `/etc/default/login`). You can do this locally through the User Manager for Domains tool by choosing Policies → User Rights, then selecting "Access this computer from network" and removing Administrator from the list (i.e., select Administrator and hit Remove).

Windows 2000 uses ACLs, but only with the Windows 2000 file system called NTFS (i.e., *not* with the MS-DOS FAT file system). Every file or object in an NTFS can have an associated ACL detailing who can access the file or object and how. ACLs are as essential to security in Windows 2000 systems as they are in Solaris (refer to Chapter 1 for a description of file permissions and ACLs). Directories in Windows 2000 have default ACLs that will be applied to any subsequently created file.

One oddity in the NTFS bears mentioning. There is an ACL right called *Bypass Traverse Checking* that is given by default. This particular right allows users to bypass directories for which they do not have permission; they can enter and access files further embedded in the directory structure as long as they know the full path. Unix systems, on the other hand, prevent anyone from entering or listing the contents of directories if they do not have proper permissions on the directory itself (regardless of whether they know pathnames or whether the embedded directories are open). We advise that you disallow this right, and thereby enforce the expected blockage when a directory is locked.

NOTE As with Unix systems, the policy of least privilege holds in Windows 2000. Anyone with administrator access should be instructed to use the least privilege necessary for the task at hand. By doing so, the risk associated with any faux pas is reduced. So is the risk associated with running a malicious program or one contaminated with a virus. For those of us with enough history to admit that we make mistakes, the least-privilege policy is a matter of professionalism, not a questioning of our status or competence.

Other groups in the Windows 2000 system can be given the privilege required for many administrative tasks. For example, the Backup Operators, Server, and Account groups can be given privileges so that full administrator privilege is not required by the members of these groups to perform the given tasks. Administrative groups can be created on the domain, thereby providing privileges at a network level.

One of the most critical elements of Windows systems that you will need to become familiar with is the *registry.* A large and complex tree-structured data store for native as well as third-party applications, the registry in Windows 2000 systems is implemented as a hierarchy of files in Windows 2000 and is modified with the `regedit` (`regedt32`) command. Although the registry is binary (i.e., you look at its content through tools like `regedit`), it is actually stored as a hierarchy of files. Modifications to the registry should always be made cautiously, because improper registry settings can keep a system from booting. Fortunately, it is possible to both create backup copies of the registry and replace the damaged original as needed. It is important to note that, in spite of its centrally important role on Windows 2000 systems, the registry is not the central repository of *all* application parameters. Some important services, such as IIS, store some of their configuration information in their own binary files and some in the registry.

Unlike Unix file systems, NTFS does not have a single root. There is a root (i.e., \directory) on each disk, and you switch from one disk to the other by entering the drive tag (such as D:). This does not imply, however, that there isn't a strong analogy between the system directory on Windows 2000 systems and the root and `/usr` directories under Unix. Most of the directories within the WINNT directory structure are protected against modification by the general user. New (postinstallation) files are generally *not.* You should be cautious, therefore, when you install application software that important files cannot be modified or removed by users. You cannot depend on third-party providers—especially if they build for both the (security weak) Windows 95/98 and the Windows 2000 platforms—to do this for you.

Windows 2000 developers seem to lack a general consensus about where and how application software makes updates within the `WINNT` and `WINNT\system32` directories. Solaris applications primarily arrive as packages expecting to be installed in `/opt`. The only impact the installation will have on our root file systems is to add, as required, start and stop scripts in the `/etc/rc?.d` and `/etc/init.d` directories. In addition, proper permissions are established during the installation. Windows applications aren't as well behaved. You will find an assortment of application files in the `/WINNT/system32` directory of any well-used Windows 2000 system. Considerable attention should be paid to securing Windows 2000 if it is to be made as secure as you expect your Solaris systems to be out of the box.

The Command Line versus the GUI

Those of us who are long-time Unix sysadmins have a decided preference for the command line. It's not because we're old dogs unwilling to learn new tricks. It's not because we're technodweebs. We *know* that, in spite of the ease with which many things can be done using modern GUI tools (such as `admintool`), most of the things we need to do are more easily done in scripts in which we can also incorporate related tasks or with which we can accomplish the same task for an entire group of systems or users. Adding a user or two through `admintool` is easy; adding 100 is painful. A simple script and a single `foreach` command can be done with the job in less time than it would take us to add the first user with `admintool`. We also know that we don't always have a GUI available. If Sandra wants to add a user while logged in from the laptop on her sailboat and anchored off Angel Island, she's going to use the command line. If Evan wants to check the temperature of his CPUs while on a business trip to New York, he's going to use the command line.

Every version of Microsoft Windows *has* a command line, of course. The problem is that Windows (even Windows 2000) does not provide the same control options on the command line that are available using the GUI. While Windows 2000 is a dramatic improvement over NT 4.0, there are still gaps. Take, for example, the issue of links. Whereas Unix systems have hard links and symbolic (soft) links, and the *ln* command to create either, Windows has shortcuts. Similar in function to symbolic links, shortcuts can be created only using the Windows GUI. This is one of many things that one must consider when trying to remotely manage a Windows system or manage a group of Windows systems. If you want a shortcut, you need to visit the system in person, coax the user to create one, or use a tool like pcANYWHERE or the free VNC from AT&T Research UK.

This issue isn't a small one. The more Windows systems you support, the more you will wish that you could Telnet into these clients and issue commands as easily as you can in Solaris. Tools like pcANYWHERE and SMS remote control (provided with the Microsoft SMS package) provide for remote control of the Windows interface, but what can you do if pcANYWHERE or the SMS agent isn't running? Ironically, you can't start pcANYWHERE and put it in to "be a host" (i.e., allow another system to remotely control me) mode from the command line. One of the authors recently drove more than 100 miles to click a button and set up this remote control feature.

Fortunately, the situation is not as dire as we, being long-time Sun users, first imagined. Given some features available, though not obviously so, with Windows 2000 plus the Windows 2000 Resource Kit) and some third-party tools, it is possible to approach the level of control that we have with our Unix systems.

One of the first things you need is a Telnet server. If all your Microsoft servers are running Windows 2000, you are in luck. The Telnet service comes standard and can be enabled simply by starting the service located under the Control Panel → Administrative Tools → Services. If you have the misfortune to have older NT servers, you need to equip every server with a Telnet service. A Telnet service is available with the NT Resource Kit or, inexpensively, from a number of vendors. Once you have a Telnet service installed on the servers you need to administer, the need for making the rounds is dramatically reduced.

There are some commands that are available from the command line in Windows 2000 natively—and we are grateful for each and every one of them. Table 20.1 shows some of them and their nearest matches in Solaris.

Table 20.1 Solaris Commands and Windows Native Equivalents

SOLARIS	WINDOWS
`share /usr/local`	`net share newshare=D:\apps`
`mount remhost: /usr/local /usr/local`	`net use h:\\remhost\apps`
`cat /etc/passwd`	`net users`
`passwd sandra`	`net user sandra *`
`adduser evan`	`net user evan password /ADD`
`ypcat group`	`net localgroup`
`/etc/init.d/httpd start`	`net start "World Wide Web Publishing Service"`
`kill PID`	`net stop "Service Name"`
`chmod and chgrp`	`cacls`
`fsck /apps`	`chkdsk D:`
`diff file1 file2`	`fc file1 file2`
`grep`	`find`
`vi file`	`edit file.txt`

As you can see from Table 20.1, this set of commands—and any we might have missed—is extremely limited compared to what you can do in Solaris. In addition, the list is missing some very notable entries—such as reboot. This situation is much improved when a copy of the Windows 2000 Resource Kit (4.0 or later) installed. The resource kit includes a lot of tools and commands that would have made Windows 2000 a more complete and usable OS. In addition, you will find collections of tools on the Web that allow you to use what appear to be Unix commands, though don't be surprised if they don't work quite like their counterparts.

Some of the commands available for remotely managing Windows 2000 systems may not work as expected. In the following excerpt, one of the authors was trying to restart IIS without success. After a hearty try, a call was made, and one of the urchins at the remote facility kindly went and power-cycled the box for us. How much more productive the morning would have been if the initial commands had done the trick! We've been told by folks who live with Windows 2000 servers more intimately than we do that the servers like to be rebooted now and then. The system in question had been up for about three months.

```
C:\INETPUB\www;>net stop "IIS Admin Service"
The following services are dependent on the IIS Admin Service service.
Stopping the IIS Admin Service service will also stop these services.
```

```
   FTP Publishing Service
   World Wide Web Publishing Service
Do you want to continue this operation? (Y/N) [N]: Y
The requested pause or stop is not valid for this service.
More help is available by typing NET HELPMSG 2191.
C:\INETPUB\www>net helpmsg 2191
The requested pause or stop is not valid for this service.
EXPLANATION
This command is invalid for this service, or the service cannot accept
the command right now.
```

Some important network troubleshooting commands are available for Windows—some through the Resource Kit. These include `ping`, traceroute (`tracert`), `ftp`, `netstat`, `route`, `rsh`, `rcp`, `rexec`, `rcp`, and the `at` command. Though most Windows users don't know or care about many of these commands, you very well might. Most, if not all, zip programs can run from the command line, making installation of software or files easier. In addition, you should consider acquiring some great scripting tools—such as Perl, Expect, and Tcl, which are available free from the Web. Adding these tools might help you take control of these clients in a manner that is more consistent with your typical system management style.

TIP **Run Windows 2000 or XP, not Windows 95/98, on all the Windows systems you are responsible for managing. Not only is it a more stable operating system, it is much more suited for use in management of other systems.**

You should set up `rsh`, preferably on your primary domain controller (PDC), though any domain controller will do. If you configure it to accept connections as root from a particular Unix machine, you can use `rsh` to change passwords on your Windows 2000 clients. The Windows version of the `rhosts` file should be installed as `\winnt\system32\drivers\etc\.rhosts`.

NOTE **Your `rsh` should be running as Administrator or a user with account creation privileges and granted the "log on as a service" right. A useful account-synchronizing script, written in Perl and using Expect, is available from `www.cis.ksu.edu/~mikhail/Passwd/passwd.tar`.**

The syntax for the `rsh` and `rexec` commands is as follows:

```
rsh host [-l username] [-n] command
```

and

```
rexec host [-l username] [-n] command
```

where the `username` argument is the login ID of the user.

Remote Management Options

At first blush, it looks like Unix systems can be remotely managed, and Windows systems cannot. Tools such as `rdist` and NIS/NIS+ provide us with options to determine how we want to reduce the task of managing hundreds, maybe even thousands, of systems to issuing changes that should be applied to groups of systems. We also have scripting languages and `cron` facilities for automating routine work. The ideal administration console allows you to perform all administrative tasks from one location and most administrative tasks with little or no duplication of effort. We'd like to say "for all users . . ." or "for each client . . ." and have our tools be adequately sensitive to the underlying differences between user environments and hardware/software platforms to do the right thing.

After spending some time exploring the options for remote management of Windows 2000 systems, we were encouraged. For one thing, we discovered that Windows 2000 is not as badly constructed as we had imagined. In fact, Windows 2000 has many remote access primitives that can be used effectively—if you're willing to code in C or C++ and use system calls. For those of us less inclined toward systems programming, there are still options well worth considering.

Because automation is the crux of systems administration, we consider it next to essential that you equip your Windows 2000 systems with Perl. By doing so, you will have a modern, powerful scripting language on both your Unix and Windows 2000 systems (we *assume* you have and use Perl on your Solaris boxes). With Windows 2000, Perl, and an enabled Telnet daemon, your Windows 2000 boxes will be ready to be managed. You will be able to manage accounts and services, distribute software, make changes to the registry, and reboot your Windows 2000 systems from afar. You will also be able to start and stop services and determine what services are running.

Networking Issues

Most Ethernet connections today are 10-baseT and 100-baseT. Newer Sun systems and PCs will often be equipped with the capability to run at 10, 100, or even 1000 megabits per second (gigabit Ethernet). Yet, even while they can easily share the same wire, there are some vastly differing technologies available when it comes to Windows 2000 systems. Windows 2000 systems have multiple network stacks available. These include NetBEUI, IPX, and TCP/IP. When a Windows 2000 system is set up, a decision must be made about which of the protocol stacks to use. Some stacks, for example, NetBEUI and TCP/IP, can be used simultaneously (i.e., they don't compete, they respond to different network traffic). Our advice, however, is to forget NetBEUI, IPX, and any other Microsoft networking option and use TCP/IP. In the days when Windows clients were gathered into small networks by themselves, it made sense to run whatever network protocol seemed most natural. In today's networks, this isn't the case, and compatibility with the rest of your network is a far more worthy goal. Network settings on NT systems are made through the control panel, using the Control Panel → Network → Adapters sequence of menu choices. In Windows 2000 it is under Settings → Network and Dial-Up Connections.

When managing Windows systems, you need to decide whether to use fixed IP addresses or to assign addresses using DHCP or bootp. Either service can be provided on a Sun server and will work reliably. Avoid mixing these two strategies (fixed and assigned addresses) if you can. You don't want to run into situations where a DHCP host acquires an IP address and then finds itself conflicting with a system that claims the same address as its permanent address. If you must mix strategies, be very careful to identify the range of addresses that will be assigned dynamically and another range to be used for hosts whose addresses are assigned on a permanent basis.

We suggest using DHCP for all of your Windows clients. It's simple to set up and could be especially helpful if some of your users bring laptops into the office—their network settings will not have to change as they change locations; they will always acquire a local IP address.

Name Services

NIS+ is available on Linux, but not on Macintosh and Windows clients (as far as we know, anyway). Similarly, Linux systems will happily run `ypbind` and talk to your NIS servers. Macintosh and Windows systems probably won't. We suggest that you set up your Linux clients to use NIS or NIS+, and specify DNS for your Macs and PCs. This strategy won't help with usernames and passwords, of course, but users of these systems are not likely to play musical computers, and if they do, they may limit themselves to working on similar systems. In this case, you can take advantage of the User Manager for Domains tool in Windows NT or the Active Directory Users and Computers tool in Windows 2000 to centrally manage your Windows users.

The Windows Internet Naming Service (WINS) will neither recognize Unix systems nor interoperate with DNS. The option of providing a WINS/DNS gateway is possible, but probably less effective than simply using DNS. If you are running an all Windows 2000 server and client network, there is no need to run WINS as Active Directory (an LDAP-based Directory service), and DNS will be all you need. If you can't easily provide information to your Windows clients using DNS, you can put essential hosts in the LMHOSTS file. This file is roughly the equivalent of the `/etc/hosts` file on Unix systems.

File Serving

Solaris and Windows systems use incompatible technologies for sharing files across systems. Solaris uses NFS, of course. Windows systems (both NT/2000 and 95/98/ME) use the Server Message Block (SMB) protocol. To circumvent this incompatibility, you can install NFS on your Windows clients or you can install SMB on your Solaris servers.

There probably aren't many platforms in existence on which you *cannot* run NFS. PC-NFS may have come and gone, but other products are available and more than fill the void:

- *DiskAccess,* an NFS client product from Intergraph, is included in Microsoft's Windows NT Option Pack for Solaris. Intergraph also provides NFS server software with the (purchasable) DiskShare software. Setting up Windows systems as NFS servers will allow you to access their file systems remotely.

- NetManage offers *Chameleon NFS.*
- Another product, called Unix Services for NT formerly called *open-NT* (nee Interix), provides a full Unix subsystem withinWindows 2000. Microsoft sells and fully supports this add-on product.
- Network Appliance and other black-box manufacturers provide *nFS* and *CIFS*-network-based storage that is accessible by both platforms.
- *PC-netlink* is a Sun product that will provide native SMB services and NetBIOS over IP services for Windows clients on a Sun Server. This derives from the Windows 2000 server code that was licensed to AT&T from Microsoft. PC-netlink is described more fully in the next section.
- *Samba* is server software for Unix (e.g., Solaris and Linux) that provides network file and print services for clients using some variants of SMB protocol. SMB is a native networking protocol used by MS-DOS-based (in a very broad sense, including derivatives) clients. Samba, which is now provided free of charge with Solaris 9, or available as freeware for older Solaris versions, is described in more detail later in this chapter.
- Hummingbird and other manufacturers provide NFS gateway services (e.g., *Maestro* from Hummingbird) that allows access to NFS resources via the native Windows 2000 service.

There are probably many other products, as well. We suggest that you refer to the PC-Mac TCP/IP & NFS FAQ List managed by Rawn Shah for additional possibilities and details at www.rtd.com/pcnfsfaq/faq.html.

Solaris PC Netlink

Solaris PC NetLink is software that provides Windows NT services on a Solaris server. This includes Windows NT 4.0 Primary Domain Controller (PDC), Backup Domain Controller (BDC), Member Service functionality, Windows NT file system support (NTFS), and Windows printing support. In other words, these services can be used to replace NT servers providing any of these functions. Tools such as User Manager for Domains and Server Manager work just as native NT services.

PC Netlink, from Sun Microsystems, Inc., provides native NT 4.0 network services on a Solaris system. Originally code named Project Cascade, PC-netlink is based on the AT&T Advanced Server for Unix product. PC Netlink offers Windows 2000 file, print, directory, and security services to Windows 3.11, 95/98, and NT and Windows 2000 clients. In fact, there are no modifications required on the clients; from their perspective the PC Netlink server appears as just another NT/2000 server. Installation is simplified with InstallShield wizards. Once installed, management of PC Netlink is accomplished via the command line or with a client/server GUI-based Admintool. Administration of the NT network services is accomplished with the native Windows server management tools from a Windows client.

In operation, the PC Netlink server works as a primary domain controller (PDC) or backup domain controller (BDC). The NT domain trust model is supported with full, bidirectional trusts, one-way trusts, or multidomain trust relationships. As of this writing, Sun is promising the ability to operate as a member server—that is, to provide file and print services without the need to be a domain controller. In addition to the Solaris-based directory services (NIS, NIS+, DNS, and LDAP), PC Netlink implements NetBIOS and WINS naming services.

The NT security model is implemented with support of the Security Account Manager (SAM), Security Identifiers (SIDs), and ACLs. Users in a domain do not get automatic access to native Solaris resources. For example, a user cannot log in to the Solaris side unless this functionality is specifically configured.

There is no automatic mapping of NT and Solaris user accounts; tools to migrate accounts (SID and Unix ID) are provided, however, and must be run as root.

The Windows clients perceive the shares on the PC Netlink server as NT File Systems (NTFSs), with the NT file security automatically maintained. Printers can be Solaris or network attached and shared between the Solaris and NT/2000 environments.

PC Netlink is bundled with many of the Sun servers (Ultra2 through E3500) and is included in the Solaris Easy Access Server 3.0. Sun provides a Sizing Guide on its Web site at www.sun.com/interoperability/netlink/whitepaper/sizing-guide/.

Samba

An alternative to installing NFS services on your clients is to provide SMB services on your Solaris server. Samba software is included in Solaris 9, but can be easily added in other versions. The free software Samba centralizes the installation and administration of your file sharing by supporting SMB.

You can get Samba from http://samba.anu.edu.au/samba. Samba installation and setup is trivial; Samba is configured as a Solaris package. The following listing depicts an excerpt from a Samba configuration file (`smb.conf`) specifying the equivalent of an `/etc/dfs/dfstab` entry for a file system being made available via Samba.

```
; Configuration file for smbd.
; ======================================================
; For format of this file, please refer to man page for
; smb.conf(5).
;
[global]
   security = user
   workgroup = FOO
   os level = 34
   preferred master = yes
   domain master = yes
   domain logons = yes
   logon script = %U.bat
   guest account = nobody
;  Global Printer settings
   printing = bsd
   printcap name = /usr/local/lib/samba/lib/printcap
   load printers = yes
;  This option sets a separate log file for each client
;  Remove it if you want a combined log file.
   log file = /spool/samba/log.%m
```

```
[netlogon]
   path = /spool/samba/netlogon
   public = yes
   writeable = no
   guest ok = yes
;  You will need a world readable lock directory and "share modes=yes"
;  if you want to support the file sharing modes for multiple users
;  of the same files
;  lock directory = /usr/local/samba/var/locks
;  share modes = yes
[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mode = 002
[printers]
   comment = All Printers
   path=/spool/samba/lp/
   print command=/usr/bin/dos2unix -ascii %s | /bin/lp -d %p -s |
rm %s
   browseable = no
   printable = yes
   public = yes
   writable = no
   create mode = 0700
;
; File system for PC applications
;
[apps]
   comment = PC Applications
   path = /apps
   read only = no
   valid users = mac pcleddy
```

NOTE Unlike SMB on Windows clients, Samba provides unidirectional (i.e., server-side) file sharing. Samba does not provide access to the clients' file systems.

In a mixed Windows network, an NT or Windows 2000 server defines the network domain and serves as the PDC.

Users on that system and others are authenticated by the PDC, while logins are authenticated by the PDC's Security Account Manager (SAM)—username, password, and domain—database.

Another service provided by Windows NT/2000 is *browsing.* An NT/2000 server can be configured as a domain master browser. In this role, it allows clients to search for available resources and printers. On the user's desktop, these resources are viewed through Network Neighborhood or the Windows Explorer shell.

It is possible to use Samba as a replacement for an NT/2000 server. It can act as the primary domain controller and as the master browser. Instead of authenticating your Windows users via a Windows server, you can allow Samba to provide this service using your `/etc/passwd` file or `passwd` map.

Samba uses two daemon processes to support its services: `smbd` and `nmbd`. The `smbd` process provides file and printer support using the SMB protocol (as its name suggests). The `nmbd` process provides NetBIOS support to the clients. These services are generally started with a startup script, as shown here. This particular file is installed as `/etc/rc2.d/S99samba` and `/etc/init.d/samba`.

```
#!/bin/sh
#
# Default SAMBA initialization/shutdown script.
#
ECHO=echo
SAMBADIR=/opt/samba
case $1 in
    start)
      $ECHO "samba: \c"
        $SAMBADIR/bin/nmbd
  $ECHO "nmbd \c"
#        $SAMBADIR/bin/smbd -p 177 -s /etc/samba/smb.conf
         $SAMBADIR/bin/smbd -s /etc/samba/smb.conf
      $ECHO smbd
  ;;
    stop)
  kill 'ps -e | grep smbd | awk '{print $1}''
  kill 'ps -e | grep nmbd | awk '{print $1}''
  ;;
    restart)
  kill -HUP 'ps -e | grep smbd | awk '{print $1}''
  kill -HUP 'ps -e | grep nmbd | awk '{print $1}''
        ;;
esac
```

NOTE Two things that you need to watch out for in using Samba are how it participates with other domain controllers in the master browser election and whether you want it to serve as PDC. Pay careful attention to the lines in the configuration file (`smb.conf`) that determine whether Samba should attempt to play these roles. A sample `smb.conf` file is shown earlier in this chapter.

Managing File Types

One of the problems that you'll encounter in sharing files in a mixed client environment is the differing line-termination conventions that each of the three client populations uses. Unix systems (including Linux) use only a new-line (or linefeed) character: octal

12. Windows-based systems use both a new line and a carriage return: octal 12 and octal 15. Macintosh systems use only the carriage return: octal 15. If your systems share files, you need to do something about these conventions.

One technique is to use FTP or some version thereof, such as `fetch`, to transfer files from server to client and back as needed. The reason that FTP has different modes, such as text and binary, is that it handles the data differently. When you specify *binary,* FTP transfers a file byte by byte with no translation whatsoever. The copy on the client, therefore, is an exact copy of the file on the server. When you specify *text* (this is usually the default), FTP *translates* the line endings. This means that the file is proper for the architecture of the receiving client.

Conversely, if you use NFS (or Samba) to mount server file systems on clients, all clients use the files *in place*; that is, they work on the file as it exists on the server. This can lead to some interesting results. Barber-pole output such as the following occurs when a file with only new-line characters (i.e., a Unix file) is displayed on a system expecting new-line and carriage-return characters (i.e., a Windows or DOS system):

```
Oh, ye tak the high road and I'll tak the low road,
                                                   And I'll be
in Scotland afore ye,
                     For me and my true love will
                                                   never meet a
gain on the bonnie, bonnie banks of Loch Lomond.
```

The *^M* syndrome occurs when a file with both new-line and carriage-return characters (i.e., a Windows-based system) is displayed on a system that expects only new lines (i.e., Unix):

```
Oh, ye tak the high road and I'll tak the low road,^M
And I'll be in Scotland afore ye,^M
For me and my true love will never meet again^M
on the bonnie, bonnie banks of Loch Lomond.^M
```

One of the consequences of managing a mixed environment is this annoying problem. The solutions:

1. Use FTP (or a derivative tool).
2. Provide translation tools.

Solaris provides two appropriately named commands for translating text files from Unix to DOS and back again—`unix2dos` and `dos2unix`. The `dos2unix` function could also be accomplished with a command of this nature, which removes the carriage returns:

```
cat file | tr -d "\015"
```

A `mac2unix` filter would look something like this:

```
cat file | tr "\015" "\012".
```

Binaries are, of course, a completely different issue. The best advice we can offer about binaries is to clearly differentiate them. Any Windows or Macintosh binaries that you keep on your servers should be in directories that are clearly named. Most sites don't maintain these binaries on their Sun servers. Disk space on clients is large enough to accommodate all the applications that these clients are likely to need. Of course, this means that updating these clients is a big deal. You might use server partitions to provide the means to easily update clients. Installing clients from a single mounted CD-ROM is probably the best of all possible worlds.

Printing

Sharing printers between Solaris and Windows systems can be accomplished in one of two ways. You can set up your Windows clients to print to TCP/IP printers. This is probably the easiest and most flexible way to share printers. You can also use Samba to print via the SMB protocol. If you have printers that do not have IP stacks, this may be the only option available. The `smb.conf` excerpt shown earlier in this chapter includes a printer-sharing specification.

Generally, you can set up a Solaris system or a Windows 2000 server to act as a spooler for a specific printer. Whatever system you choose, consider the space required for spooling large files along with the number of people sharing the printer.

Directory Services

Where and how you manage user accounts—in your Windows 2000 domains or your NIS domains—is an important decision. Your users can exist in both domains and might need to work in both. If they do, you'll have to set them up twice—having the same username and password in each environment will make this easier on them, but less desirable from a security point of view.

Windows 2000 systems have PDCs and BDCs that function in roughly the same way as master and slave NIS servers. In both systems, changes should be made only on the primary or master server and then propagated to backups or slaves.

In Windows, a system must first be registered on the PDC to become a member of the domain. This is not true on Solaris, of course, where a client can bind to an NIS domain without being included in the hosts map.

Email

One of the universally important services on any network is email. A cornerstone of today's business model, prompt and proper delivery of email is essential to a healthy network. There are a number of approaches with respect to your non-Unix clients:

- POP clients
- IMAP clients
- Web-based email solutions
- Unix character-based tools

If you're using the X Window system on clients, you can, presumably, also run X-based tools to read and send email from client systems. Of the other options, POP and IMAP are probably the most common choices. Most client mail tools (Eudora, Outlook, Outlook Express, etc.) allow either protocol to be used. As mentioned in the previous chapter, the option of whether to leave mail on the server is critical. For users who only use mail from a single system, the download option (i.e., *not* to leave mail on the server) is probably the best. The client software will fetch mail from the server, erasing it. For users who read email from work, from home, and while on the road, this option potentially leaves different sets of email messages on each of three different systems. This could make this option extremely difficult to manage and frustrating to the user, as well.

Users can log in to your Unix servers and run a mail tool through their Telnet or SSH session—such as elm or pine. On the other hand, users accustomed to more user-friendly front ends might find this approach less than desirable.

Other Approaches to Compatibility

Other approaches to providing interoperability in the heterogeneous network involve using tools that don't care about the differences in the underlying hardware and operating systems.

The X Window System

One of the earliest and most effective solutions to the compatibility problems between Unix, Windows, and Macintosh incompatibilities is the X Window system. What X allows you to do is run an application on the host for which it was written (say, a Sun Enterprise file server) while controlling it (via the X server) on the client. Keep in mind here that the X terminology reverses what one normally thinks of a server and client in these arrangements. In the X vernacular, the server is the system in front of a user that is displaying windows, while the client is the software running in any particular window.

The Web

A newer and nearly ubiquitous platform for providing tools across a diverse population of clients is the Web. Any tool built to run in a browser, from a mail tool to a database, is instantly available on any system that has one. Many wise corporations are realizing that their porting days are nearly over. True, there are still compatibility problems. Netscape and Internet Explorer behave differently in some respects. Still, as a means of providing an intranet—and a wide collection of tools and information repositories across the organization—the Web is hard to beat.

A Web-related technology that is also providing application compatibility across different systems is Java.

Client/Server Applications

As nice as the Web is, it is not the ideal platform for just any application. Tools for dynamic and interactive content on the Web are still evolving, and any tool that runs through a browser is encumbered with some degree of overhead. Applications that run in a true client/server mode—in which heavy-duty back-end processing is handled on the server, with the front-end user-interaction portion of the work running on the client—are often built to run in today's typical heterogeneous network. The server software often requires the robustness and processing speed of a Sun system, while the client end uses the resources of a Windows client and doesn't affect users on other systems when individual users shut their system down or find themselves staring at the blue screen of death.

Accessing Clients

You cannot remotely log in to most Windows clients. Unlike Unix systems, there is no Telnet daemon available to provide you with remote login. This is another reason that these clients are both difficult and time-consuming to manage. You can purchase Telnet software for Windows 2000. Windows 2000 servers also provide an FTP service (i.e., you can ftp *to* them, not just *from* them). There is very little difference from a functional standpoint between a Windows 2000 server and a Windows 2000 host. You might insist that more critical Windows systems provide you with the means to log in and install files from your desk.

Porting Your Expertise

One of the biggest compatibility issues that many of us have to face when it comes to the Windows and Macintosh clients making their way into our offices and server rooms is not a technology issue at all, but our own readiness to deal with these systems. Many organizations have different staffs to support Unix systems and PC/Mac clients. Even so, you will find yourself in many frustrating situations if you don't know enough about the way the non-Unix systems work to be able to blame them or rule them out when some element of your interoperational network doesn't work properly.

If users come to you and say they can no longer retrieve mail from your mail server, what should you do? You can suggest that they double-check the settings in their mail tool (e.g., their POP server, username, and password), but they may just look at you and claim they haven't changed a thing. Another test is to Telnet to the POP port on the server (i.e., `telnet mailserver 110`) and have them try their name and password right then and there. If they log in successfully, you'll know that your mail server as well as their login information is OK. This is just one situation. There will be many. The problem with separate staffs managing separate problem areas is that there is often no one bridging the gap enough to determine which domain a particular problem lies in.

We suggest that you know enough about the non-Unix platforms to be conversant in the tools. You should understand the operations of and the way to make changes to the registry on Windows systems. You should understand the system folder on a Macintosh. You should know where basic networking information is configured. This includes the following information:

- The host name
- The IP address
- The network mask and router

- The name server
- The domain

You should also understand how to configure mail servers (and, hopefully, you won't have to deal with more than one or two of these since they're all different) and print services, as well as take a look at disk usage and availability.

Summary

Today's networks are heterogeneous. There seems to be a steady migration of resources. Big (and even bigger) Sun servers are moving into our computer rooms, and a mix of clients is moving into our offices and cubicles. This chapter attempts to describe some strategies for managing this mixed environment, especially for making decisions with regard to Windows systems to make them more amenable to being managed.

What we probably need most are integrated tools that make it possible to manage Windows PCs, along with other non-Unix systems, from a central system. Network management tools with agents that run on these clients may provide some of this control, but at a hefty price tag. In the interim, a careful sharing of resources may start the evolution toward a manageable coexistence.

We discovered, in our struggles with taming out-of-the-box Windows 2000 systems, that the only hope of making them manageable is to add a collection of tools. The Windows 2000 Resource Kit is a must. Besides this, we strongly recommend that you install Perl; many versions of Perl5 are available for Windows 2000 systems. We also recommend that you look into using pcANYWHERE or VNC to allow you to remotely take over control of the GUI when remote options don't cut it.

Learning to manage Windows 2000 systems as well as you can your Solaris systems is going to be a learning experience. In time, we expect tools will emerge that will take advantage of the Windows 2000 primitives and hand us the steering wheel. In the interim, beef up your toolkit and be prepared for some hard work.

The most encouraging change from our users' perspective is undoubtedly the introduction of GNOME as an alternate desktop for Solaris. The GNOME desktop provides a suite of applications in what we believe is the most easy to manage desktop environment available today. In the mixed Linux/Solaris environment, the introduction of GNOME into Solaris promises something of a return to homogeneity.

CHAPTER 21

Time Management Techniques for Sysadmins

"A system administrator's work is never done"—the phrase once used to describe mothers describes the "mothers" of today's busy networks. How many of us work 8-hour days and leave our jobs behind when we're at home with our families or lounging with friends? Very few, we suspect. The authors have spent too many years on the job not to recognize how deeply our jobs permeate our lives and our identities.

Still, there are steps that we can take to better control how much our jobs dominate our lives and to manage our work responsibilities so that they don't control us. This chapter offers a number of suggestions that have proven themselves valuable in our careers and just might be useful in yours.

Daily Priorities

One of the most effective techniques that we use on a daily basis is to prioritize our daily tasks before we do anything else. We start each morning with a list of the tasks we intend to work on that day and use this list as a checklist as we accomplish each task and move on to the next. Anything short of an all-out emergency waits until we have roughed out this list of what we need to work on and where each task falls into our queue of things to do.

To the extent that we can easily communicate this list to our users, we will find ourselves with a more understanding customer base. It is, unfortunately, very easy for any individual to think of his or her demands on your time as being the only thing you have to do. Unless you actively and accurately communicate the variety of tasks that you are juggling at any point in time, these individuals are likely to become impatient when their requests are not completed as quickly as they imagine is possible.

Managing Priorities

Establishing your priorities can be a tricky matter. There are many factors to consider when looking at any particular problem or task—the number of people affected, the degree to which they are incapacitated or inconvenienced because the work is not done, and the length of time and difficulty that the task involves.

Developing a reasonable prioritization methodology is probably the most important aspect of setting priorities. Whether you first manage tasks that have the most impact or those that are can be quickly "knocked off" might be an issue to discuss with your boss or to communicate to your user base. There are clear choices to be made. One of the authors, for example, insists on sticking to a prioritization model in which the highest-priority tasks are accomplished first. The other, on the other hand, allows low-priority tasks that are trivial in both complexity and time required to "slip through" to the top of the list—the idea being that any task takes less time if you simply do it while you're thinking about it.

Deadlines for Everything

A good practice in managing tasks is to give everything that you have to do a deadline. By doing this, very few tasks will get pushed back indefinitely. Some sysadmins devise a scheme, either formally or informally, for elevating tasks in priority as they sit in their work queue awaiting attention. Any tasks that remains undone past the initial deadline that you set should be considered high priority unless you are ready and willing to determine that your initial deadline was unreasonable and assign it a new one. You don't have to be correct in your estimates of how long a task will take; you simply have to devise a mechanism by which you can keep track of tasks (i.e., so that nothing falls through the cracks) and continually revise your tasks so that your priorities are rational and easy to communicate.

A Time for the Routine

Everyone has a certain amount of routine, boring, inane, innocuous, and otherwise "low-tech" work to do. The best advice that we can give you on getting this work done without excessive or insufficient attention is to schedule this work for a specific time of day. You might, for example, deal with this type of work every day right after lunch.

Continuous Inquiry

Another common practice that we use in managing our tasks is a low level but almost continuous review of what we're doing. Even with a written priority list, it's good to ask yourself from time to time "What is the best use of my time right now?" Numerous items on your priority list may be delayed for technical or management reasons. To work on anything when progress is unlikely or minimal is to waste your time. Maintain a healthy cynicism. Never allow pressure to wind you into a tight circle of useless activity in which no progress is possible.

Expect to Lose 10 Percent

Never plan to spend 100 percent of your workday on your scheduled tasks. Inevitably, there will be interruptions. Plan to spend 90 percent or less of your time on planned tasks and for the other 10 percent or more to be spent on tasks which you didn't anticipate or on "changing gears"—the basic human need to regroup intellectually as you finish one task and prepare to take on the next.

Managing Phone Time

Scheduling time for phone calls will help to ensure that necessary calls are placed, while keeping you from the distraction associated with phone calls during the bulk of your day. We suggest that you schedule calls before 10 A.M. and after 4 P.M. each day. One of these time periods or the other will work fine for calls you need to make if you're on one coast and need to call the other.

Don't spend any time on hold. Your time is far too valuable for this.

Handling Mail—Email, Real Mail, and Junk Mail

Sort your mail by priority. Toss out junk mail unopened. Hold questionably interesting mail in folders by date and toss it if you haven't managed to process it within a week or two. Make a firm policy of handling each piece of mail once and once only. If it doesn't pertain to something that can be acted on without a second perusal, it probably isn't worth saving. Be harsh. You may be pleasantly surprised by the weight lifted from your shoulders when there is little unfinished business piling up on your desk.

Making Meetings Count

To the extent possible, use conference calls in place of physical meetings. The time you save just in waiting for meetings to start can be used to get work done.

TIP When you have to meet in person, schedule meetings right before lunch or at the end of the day. Somehow, this shortens them without any additional effort.

Immediately after a meeting, if not during, make a list of those action items that you need to handle. While they're still "warm," prioritize these action items and annotate your to-do list to reflect these priorities. At any other time, remembering and prioritizing these tasks will take twice as much time or longer.

Taking Notes

Never rely on your memory. Write everything down. Never rely on scraps of paper and backs of envelopes. Write everything down in a place where you're not likely to lose track of it. Recreating your notes and ideas can take an inordinate amount of time.

As you finish a task, no matter how small, make a note of it. By the end of each week, you will have a list of accomplishments. Each of us has had times when, by the end of an extremely busy and exhausting week, we could barely remember what we had actually accomplished. Even the most modest note-taking tool will help you maintain a sense of accomplishment and serve as a source of data should you ever need a sketchy but fairly representative description of how you spend your work week.

Listen to Your Body

One of the most effective policies that each of us has adopted involves knowing our "rhythms." There are times when even hard work seems easy—when our juices are flowing, when we feel energetic, and when we seem to be able to accomplish a great deal more than usual. We like to refer to these periods of time as our personal "prime time." Try to identify your own prime time—for many of us, it's that 20 percent of each day when we can get 80 percent of our work done. For some people, this is the beginning of each day. For others, it may be mid afternoon. Once you know your personal prime time, schedule those tasks requiring the most focus to coincide with it. Conversely, use times when you're running at half throttle to handle the simple, brainless tasks that pile up on your plate—filling out forms, returning routine phone calls, and sorting through your mail.

Tasks of a Feather

Without becoming overly zealous, try to organize your work such that similar tasks can be accomplished together. Look for opportunities to collapse multiple projects into one. Take advantage of the common elements in your to do list to lessen the amount of work involved in individual tasks. If you're cleaning up log files on one server, doing the same on several other servers is probably only an incremental hardship. If you're creating new accounts, disabling and cleaning up space used by accounts that are no longer active will require little extra time. Overall, by "normalizing" your work, you will likely find yourself feeling much less fragmented by the end of each day.

Be Mean, Be Practical

Be very careful when building expectations among your users. It is an interesting fact of human nature that people feel the most antagonistic when outcomes differ from expectations. If a user expects that you will have a new application up and running by Wednesday morning, each hour that the project is delayed will be irritating. Learn to say "no" or, at least, learn to say "yes, but . . . ." Users who appreciate that your list of outstanding work is considerably longer than the few things they would like you to do will be less demanding and more appreciative.

Be Neat

Yes, we said "be neat." The archetypal computer geek may have a stack of papers and books that reach halfway to the ceiling, but this is no way to work. For one thing, clutter is distracting. Even if you believe yourself immune to the mess, you are likely to find yourself considerably more comfortable and able to work on whatever task is at hand when you sit at a clean work space.

If necessary, schedule a couple of hours every two or three weeks to clean your area. File the important papers and notes. Toss the rest into the trash can (or recycling bin, when possible) without mercy.

Do It Now, Do It Right

Of all the self-organizational techniques that we use to keep ourselves productive in our demanding jobs, the single most important rule is "don't procrastinate." The longer work sits in your queue, the more time it consumes. Don't ever be fooled into thinking that it doesn't take considerable effort to keep track of various outstanding tasks. If you can get a job done the first time you think about it, it takes only the time it takes. If you have to think about it a dozen times to keep it in mind, it takes much longer and contributes to a general sense of there being too much to do. Just having a to-do list with several dozen items on it can be an emotional drain.

Sandra's one-time boss, Michael "Mac" McCarthy, describes the mental state of the overbusy sysadmin as "thrashing." By the time keeping track of all you have to do consumes more of your time than getting any of it done, you're seriously underproductive and probably miserable too. We hope that some of the suggestions we've made in this chapter will keep you from ever being in this position.

Summary

Though systems administration is a difficult job with a seemingly endless stream of things to learn and problems to solve, almost any technique that you use to better manage your time will quickly increase your sense of control. The techniques described in this chapter are those that have worked best for us over the years and, to the extent that we follow them, we have better jobs and more fulfilling lives. There's nothing like going home at the end of the week with a clear sense of what we've accomplished and a honest view of what's left to be done.

Glossary

absolute pathname A complete path to a file starting from the root (/) level (e.g., `/home/nici/data/file1`). Contrast with *relative pathname*.

access To read, modify, or otherwise use a resource—for example, a file.

access control list (ACL) A structure and related mechanism for controlling who has access and what kind of access to a resource—for example, a file or a display.

account A feature providing the ability to log in and use system resources, generally through the use of a username and password, and in conjunction with a dedicated directory and established environment. This word also refers to the directory and files belonging to the user.

address A number used by systems or resources to locate other systems or resources. Important addresses include fully qualified host names, IP addresses, hardware (also called *MAC*, *Ethernet*, or *network*) addresses, and email addresses.

address mask A string of bits used to deselect (or hide) certain bits in an address so that the remaining bits select a particular portion of the available address space. In particular, subnet masks.

address resolution The method by which addresses of one type are mapped to addresses of another type. For example, fully qualified host names are mapped to IP addresses by DNS servers.

address space In general, the range of addresses possible in any addressing scheme. In particular, the range of memory locations that a CPU can refer to—limited by the size of the address register(s) and/or the addressing method used.

agent A process that collects or transfers information as part of a larger system (e.g., a message transfer agent or an SNMP agent).

alarm A reporting tool that is triggered when specific things happen on a network or system (e.g., a critical process dies). A warning message or other signal from a process, generally alerting a systems administrator to a fault or impending problem.

AnswerBook Sun's online documentation, used with OpenWindows or within a Web browser.

architecture The specific features or components of a computer or the layout of a system or network.

archive A collection of files stored so as to preserve their state at a particular point in time.

A record With respect to DNS, an address record. An A record is the most straightforward of DNS records. It simply ties a name to an IP address.

ARP Address Resolution Protocol. See *address resolution*.

ASET See *Automated Security Enhancement Tool*.

authentication The process by which a user's identity is verified. Authentication can involve usernames, passwords, special keys, and physical devices (e.g., SecurID cards).

autofs A file system type used for automatic mounting via NFS. See *automounter*.

Automated Security Enhancement Tool (ASET) Sun's tool for assessing the security posture of a system. The user specifies the level (low, medium, or high) of checking to be performed.

automounter Sun's software that automatically mounts a directory when a user requests it (e.g., with a `cd` command).

back file system With respect to caching, the source file system. Contrast with *front file system*.

background process A command or process that runs without restricting the user's control of the session. A user can put a process into the background by putting an ampersand (&) at the end of the command line or by entering Ctrl+Z and then typing **bg** after starting the process. Contrast with *foreground process*.

backup A copy of a file system or particular directories stored on tape or other media, meant to be used to restore entire file systems or specific files as needed. Backups serve as protection against data loss.

backup device A hardware device used to back up file systems, often a tape drive.

Berkeley Internet Name Daemon (BIND) The implementation of DNS that ships with Solaris.

big-endian A way of storing binary data such that the more significant bits are stored first. Contrast with *little-endian*.

binary A numbering system based on the number 2. Used extensively on computer systems because of its natural fit with two-state electronics. Each digit in a binary number is 0 (off) or 1 (on).

binding The process by which a client determines which server it is to use for specific information or support (e.g., DNS). In general, binding is in effect until the client terminates the relationship (e.g., kills or restarts `ypbind`).

blacklisting Marking a component bad so that it will not be used.

block A unit of data, as stored on a disk, that can be transferred—usually 512 bytes.

block-based The characteristic of a file system that allocates a block at a time. Contrast with *extent based*.

block map A structure that keeps track of the locations of data blocks on a disk.

block (special) device A device that transfers data in blocks.

boot The process of bringing a computer from the turned-off or halted state to full operation. Short for *bootstrap.*

boot block An 8-KB disk block containing information essential to the boot process (e.g., the location of the boot program on the disk).

boot file With respect to DNS, the file that details the zones and the servers that are authoritative for these zones. Specifically, `/etc/named.boot` or `/etc/named.conf`.

boot PROM On Sun workstations, this contains a command interpreter used for booting. See also *ID PROM*, *EEPROM*, and *NVRAM*.

boot server A server system used to boot client systems.

bootp A network service that provides boot information (e.g., IP addresses) to clients. Contrast with *DHCP*. `bootp` is an alternative to RARP for booting diskless workstations.

broadcast A packet addressing/delivery scheme by which packets are delivered to all of the hosts in a particular network or subnet.

cache A high-speed memory buffer intended to decrease access times.

cachefs Local disk-based file system used to speed up NFS performance.

caching-only server A DNS server that is not authoritative for any domain, but caches data that it receives from other servers and answers queries from this cache.

CDE See *Common Desktop Environment*.

cfsadmin A tool used to build and manage cache.

change mode Changing the permissions associated with a file in order to change access rights.

character special device A device that transfers data character by character.

child process A process started by another process and, to some degree, under its control.

chunk Smallest retrievable piece of storage from a RAID volume.

classing engine A mechanism that allows an application to query a database in order to determine how a desktop object should be handled. For example, the classing engine determines the icon used to display a file and what happens when a user double-clicks it.

client A system that uses the resources of another (i.e., the server).

client/server model A common way of organizing services and applications so that some part of the processing burden (often referred to as the *back end*) runs on a server, while another part (the *front end* or interface) runs on the client.

cluster A group of computers connected in such as way that they act as if they were a single machine with multiple CPUs. Clusters provide one approach to high availability.

command-line interface (CLI) The space into which commands are typed in a terminal window. Contrast with *GUI*.

command syntax A description of the options available for a particular command. Proper syntax describes which elements are optional and which can be repeated, and so forth.

comment Text inserted into source code, scripts, or configuration files that does not affect the way the files are used. Comments are generally intended as notes to remind anyone maintaining the files what the files are used for and why they are coded or configured in particular ways.

Common Desktop Environment (CDE) A windowing system available on Solaris, developed jointly with HP and other vendors.

compiler A special-purpose program for turning medium or high-level language, or source code, into machine code that a CPU understands.

concatenation A process by which subdisks are combined end to end to form a larger virtual disk.

contention An attempt by systems or processes to use resources at the same time.

context switching An inherent part of multiprocessing in which one process and its *context* (access to data, files, and parameters) must be stored in order for another process and its context to be loaded.

control character A character that has a special meaning when the control key is held down simultaneously with pressing it.

core dump The residue of a crashed program—the contents of memory (sometimes referred to as *core*) dumped into a file.

crash A system failure resulting from a hardware or software malfunction.

cron The Unix scheduler. `cron` uses `crontab` files (any user may have one) to detail what tasks are to be run when.

crontab files `cron`'s configuration files. Each line has the format *`m H d M D task`*, where *`m`* is the minutes after the hour, *`H`* is the hour (0-23), *`d`* is the date, *`M`* is the month, and *`D`* is the day of the week.

cylinder A set of tracks on a disk that line up horizontally.

cylinder group Consecutive cylinders grouped together into a logical unit.

daemon A process that runs in the background, listening for requests. Daemon processes generally start up at boot time. Contrast with processes that are started by `inetd`.

dataless client A client that uses its local disk only for root and swapping and depends on file servers for everything else.

degraded mode The condition in which a RAID system operates when a component has failed.

default The value that an argument or variable will have if none is specifically assigned.

delimiter A character used to separate fields in a configuration or data file.

DHCP See *Dynamic Host Configuration Protocol*.

dig An extremely valuable tool for debugging DNS. Compare with *nslookup*.

disk array One or more physical disks organized as a single logical drive.

disk-based file system A traditional file system composed of files on a hard disk.

diskfull client A client with a disk—can be dateless or standalone.

diskless client A client that has no disk and, therefore, relies on remote file servers to provide root, swap, and other file systems.

disk partition A portion of a disk separated from the rest to provide space for a particular file system or raw data.

disk quotas A system in which users are limited in how much disk space they are allowed to use. Disk quotas involve hard and soft limits along with a grace period in which, having reached a soft quota, the user is warned that file holdings must be reduced.

distinguished name A naming component used with LDAP that uniquely identifies an object.

distributed file system A file system that is spread across multiple systems.

DNS See *Domain Name System*.

domain In DNS, a portion of a name space that corresponds to a particular organization of division of an organization. With respect to the E10000, a virtual system using a portion of the E10000's resources.

Domain Name System (DNS) The distributed naming system that defines systems on the Internet. DNS is organized into root domains, subdomains, and organizational domains, all of which communicate so that any system binding to a DNS server can determine the address of any other system (provided it has a DNS record).

dot file File used to define user environments (e.g., `.cshrc` and `.profile`). Also called *hidden files*.

dump A backup of a file system created in a particular format. Dump files can be *full* (including all files) or *incremental* (including only those files modified or created since the last backup at that level). Contrast with *tar files*.

Dynamic Host Configuration Protocol (DHCP) A framework for passing configuration information to clients on a TCP/IP network. DHCP is based on `bootp`, but has the additional capability of automatically allocating reusable network addresses.

dynamic reconfiguration The process of adding and removing a system board on the fly, changing the basic configuration of a system.

EEPROM Electrically erasable programmable read-only memory. On Sun systems, the EEPROM is used to hold information about the system used in the boot process.

encapsulation The process by which Veritas Volume Manager turns an occupied disk into a Veritas volume.

environment The parameters—including search paths, aliases, and others—defined by dot files when a user logs in.

environment variables Shell variables that are passed to child shells. Typically, environment variables are used for such things as directories (e.g., `LD;usLIBRARY;usPATH` and `MAN;usPATH`). The C shell `setenv` command is used to assign values.

escape A character that removes the special meaning of the following character, allowing it to be treated simply as a character. For example, to remove the special meaning sometimes associated with `$`, you would use `\$` instead.

Ethernet A network topology that uses particular network cabling (e.g., coax or UTP) and uses carrier-sense multiple access (CSMA) to determine if packets (frames) have collided and, therefore, need to be resent.

executable file A file that can be executed. Requires both execute permission and executable contents to work properly.

expire With respect to DNS, how long a secondary will hold information about a zone without successfully updating it or confirming that the data is up to date.

exporting The process by which a file server advertises and shares file systems. Also known as *sharing.*

extent-based A disk-space allocation scheme in which multiple blocks are allocated at once to make the process more efficient.

failback Restoration of a service on the original server after a repair.

failover The process by which a failed service is rehosted on another system.

FAT With respect to Unix systems, fat is a designator applied to file systems (e.g., the fat fast Berkeley file system). With respect to Windows systems, FAT is a file system implemented typically for MS-DOS-based systems; contrast with *NTFS*.

fault tolerant The ability of a system or component to withstand failing hardware.

field In configuration files, a portion of a line that has a meaning distinct from the rest of the line. Fields are separated by delimiters.

field-replaceable unit (FRU) Any piece of hardware that is easily replaced in the field in minimal time without disruption to the rest of the network.

FIFOFS See *first in, first out file system*.

file handle A data structure used in NFS to uniquely identify a file. A *stale file handle* is one that is out of sync with the actual file.

file manager The GUI-based tool that allows users to manipulate files with drag-and-drop operators.

filename The name associated with a file. It is stored in the directory corresponding to its location in the file system.

file permissions The permission matrix associated with a file and stored in the inode. File permissions determine who can read, write, or execute the contents of the file. For directories, file permissions determine who can list the directory's contents and `cd` into the directory.

file system In Unix, the tree-structured organization of files starting from root and including local file systems as well as those mounted from remote systems.

File Transfer Protocol (FTP) The most common program for moving files between systems.

filter In general, a program that reads from standard input and writes to standard output, using pipes. For example, the `tr` command in `cat file / tr "A-Z" "a-z"` is a simple filter. Also, a program that processes input and decides whether to pass it on or reject it—as used in filtering routers.

firewall A system (hardware, software, or both) designed to prevent unauthorized access to or from a private network.

first in, first out file system A file system which processes requests in the order in which they are recieved.

foreground process A term describing a process that has control of the current terminal. No other commands can be run until the process terminates or is put into the background.

fork The method by which a process creates a new process.

format The initial preparation through which a raw disk is prepared with a basic layout allowing it to be used with a particular type of system. After formatting, disks can be partitioned and file systems can be built.

forwarders A list of addresses to which a machine should send forward requests for sites it cannot resolve.

frame With respect to Ethernet, the data transmission unit commonly referred to as a packet is more properly termed a *frame.* The Ethernet frame contains two or more levels of addressing.

front file system With respect to caching, the local copy of a file system. Contrast with *back file system*.

FRU See *field-replacable unit*.

fsck A utility used to check and repair file systems.

FTP See *File Transfer Protocol*.

fully qualified domain name A domain name or host name that contains every element in the name (e.g., `spoon.spatulacity.com`).

GID See *group ID*.

global namespace The naming of devices in a cluster, providing them with visibility across a network.

GNU Gnu's Not Unix. A tremendously useful set of utilities available for Unix systems. The GNU C compiler, for example, is one of the most popular C compilers available. Other tools include `gawk`, `gnuplot`, `gmake`, and others.

graphical user interface (GUI) A method of interacting with a computer using graphical elements and mouse control. Contrast with *command-line interface*.

group A collection of users associated with each other by virtue of being members of the group. Group membership allows file sharing.

group ID (GID) The numeric identifier for a Unix group.

GUI See *graphical user interface*.

hard limit With respect to disk quotas, the limit at which the user is prevented from creating or modifying files.

hard link A file that shares its data and its inode with another file.

header With respect to email, the addressing information at the top of a message that identifies the sender, recipient, dates, subject line, and hosts involved in the transfer.

heterogeneous network A network composed of different types of servers and clients (in other words, a typical network).

hexadecimal A numbering system based on the value 16. Hex is used on computer systems because it is easily converted to (and tightly related to) the binary numbering system and because it is terse. For example, the hex value FFFF is equal to the number 65,535 and the binary value 1111111111111111.

high-water mark A value marking the high range of acceptable values. For example, if you set a high water mark of 90 percent for a file system, you might set an alarm to go off if the file system becomes more full than that.

HINFO With respect to DNS, a record (which is hardly ever used) put in place to detail system types.

hint zone With respect to DNS, a zone used solely to find a root name server to repopulate the root cache file.

HME Hundred-megabyte Ethernet. Compare with *Lance Ethernet* (le).

home directory The directory assigned to a user when an account is established. Generally, this will be `/home/username` and will be set up with a basic set of dot files.

homogeneous network A network composed of systems of a single basic type—for example, all Suns or all Intel-based systems.

hostid A string of characters that identifies a system. For Sun systems, this is related to the hardware address.

hot-plugging Replacing hardware boards while a system is running. Also called *hot-swapping.*

hot spare A disk that is ready to be called into service without requiring a shutdown or reboot.

hot-swappable A feature of hardware that allows boards to be replaced while the system is running.

hsfs High Sierra file system used for CD-ROM file systems.

ICMP See *Internet Control Message Protocol*.

ID PROM In certain Sun systems, the PROM that contains such information as the serial number and Ethernet address.

industry standard Commonly agreed-upon procedures or interfaces adopted to encourage interoperability.

inetd The Internet daemon. `inetd` is the process that listens for requests and starts many network services in response. `inetd`'s configuration file is `/etc/inetd.conf`.

init The process (process id 1) that starts at boot time and, essentially, starts all the other processes for the target run states. See *run states*.

initialization files Another name for dot files, which, set in a user's home directory, establish paths, environment variables, and other operational characteristics.

integrity The characteristic of data or files that remain intact.

inode A data structure containing the critical parameters (i.e., metadata) describing a file (e.g., its size, location on disk, file permissions matrix, and access/update times).

input/output (I/O) Data fed to or produced by software.

interface Generally, the access provided to any process or program. A GUI is one type of interface.

internationalization The process of modifying a program so that it is usable in many geographical locations.

internet A collection of networks used as a large network. Contrast this with the more popular term, Internet.

Internet The loose connection of networks around the globe that makes international email and World Wide Web browsing possible. Based on TCP/IP, the Internet works primarily because the protocols involved are rugged enough to survive data loss, disconnects, and nearly random routing.

Internet address A 32-bit address, using dotted decimal notation, that identifies a single host.

Internet Control Message Protocol (ICMP) The control protocol of TCP/IP that manages errors and control messages. The `ping` command is the most prominent example of this protocol.

Internet Protocol (IP) Relates to the networking layer of the TCP/IP protocol.

interpreter A program that translates high-level language into executable code one line at a time. Contrast with *compiler*.

interprocess control (IPC) A method for sharing data between processes.

interrupt A signal that breaks (i.e., halts) a command or process.

interrupt request (IRQ) A signal requesting CPU attention.

I/O bound The situation in which the use of input or production of output uses most of the resources of a system or process.

IPC See *interprocess control*.

IP datagram The basic unit of transmission on TCP/IP networks.

IP network number An IP address that identifies a network. For example, the address 222.12.34.0 represents a class C network composed of hosts 222.12.34.1 through 222.12.34.254.

IP Security (IPsec) A set of protocols being defined by the IETF to support secure packet exchange at the IP layer of TCP/IP communications.

ISO 9660 An international standard for the format of CD-ROM drives. See *hsfs*.

JumpStart A Solaris installation method that uses profiles to facilitate installations of a large number of similar systems.

kernel The core of the Solaris operating system. The kernel manages the hardware and provides fundamental services (e.g., keeping track of running processes).

kernel architecture The type of kernel on a given Solaris system.

kill To stop a process by sending it a signal. The `kill` command sends signals of many types, most of which do not terminate the processes. See *signal*.

label With respect to disks, the information written on a disk by the `format` program. It describes the disk geometry, including how the partitions are laid out. With respect to DNS, one element of a domain name. Labels need only be unique at a specific point in the tree.

Lance Ethernet (usually le0) A 10Mb-per-second Ethernet interface used on many Sun systems.

latency The difference between expected and actual performance.

LDIF The LDAP Data Interchange Format. A format used for moving information in or out of an LDAP server.

Lightweight Directory Access Protocol (LDAP) A method of simplifying access to a directory service modeled after X.500.

lightweight process (LWP) A process that shares resources, thereby being less of a processing burden.

line editor An editor that allows the user to work on the current line. Contrast with *visual editor*.

link In general, a file that points to another file. Symbolic links (also called *soft links*) contain the names of the files they point to. Hard links are identical to files they are linked to.

link board A special board that can be used to link certain Sun Fire servers together, enabling them to act as one larger system.

Linux A public-domain Unix operating system, used primarily on Intel systems as an alternative to Windows. Originally developed by Linus Torvalds.

little-endian A way of storing binary data such that the less significant bits are stored first. Contrast with *big-endian*.

live upgrade Upgrading a system without taking it down for the duration of the upgrade.

load balancing Any technology that distributes load across a number of same-type components (e.g., Web traffic).

loadable kernel module Software that can be thought of as part of the kernel, but that is loaded only as needed, keeping the kernel small and efficient.

loading The act of moving a process into memory.

local Located on the system in front of the user, as opposed to somewhere else on the network.

Local file system A file system on the particular host. Contrast with network-based file systems.

localization The process of modifying a program so that it works properly in the location in which it is used.

locking The process of making a file inaccessible to most users while it is being used by one user (or a process). Locking prevents users or processes from over-writing each other's changes.

LOFS The loopback file system. Allows a file system to be used with an alternate path.

log A file in which messages are stored (e.g., errors and messages about mail transfers).

login directory The directory that a user enters when first logging in. This is almost always the user's home directory.

login shell The shell assigned to a user in the `passwd` file or map.

logging The behavior of a file system that keeps track of changes in a separate file. Logging simplifies and speeds up recovery from file system damage because the log file can be used to remake the changes.

low-water mark A lower threshold. Compare with *high-water mark*.

lpsched The command that starts the print service.

lpstat The command to view the contents of a print queue.

MAC address Another name for a hardware or Ethernet address.

magic number A number or string stored in a file (usually at the very beginning) that identifies its type. The strings and file types are stored in the file `/etc/magic`.

mail client A system that uses another (the mail server) to provide mail files and mail services.

mail server A system that receives mail on behalf of users, generally logged in on other systems.

major/minor device numbers Numbers that identify devices attached to a system and allow the system to interact with these devices.

make A command used to direct the compilation process for a program.

makefile A file used by the `make` command. Contains instructions about the compilation process.

management information base (MIB) A data structure on a managed device that determines what information can be collected from it by a management station.

manager A component in a distributed network management system that receives data from monitored systems.

man pages The standard Unix online documentation. Man pages are divided into sections (e.g., user commands, administrative commands, etc.) and are usually stored in `nroff` format.

master server With respect to DNS and NIS/NIS+, the system from which changes are made and then propagated to slaves or replicas.

maximum transmission unit (MTU) The largest transmission that can be sent over a network. Most Internet routers communicate this information so that the smallest MTU of the network segments is used for the transmission.

mean time between failures (MTBF) The average amount of time before failures occur with respect to given hardware, determined by testing or extensive reporting.

metacharacter A character that has a special meaning to Unix or particular shells, generally used for pattern matching (e.g., $ or ?).

metadata Data that describes other data. For example, the attributes of a data file are metadata.

MIB See *management information base*.

mirror A duplicate copy (e.g., of a disk) used for high availability.

mkfs Command used to create a new file system.

mounting The process of making a system available on a system. Contrast with *exporting* (sharing). See *unmount*.

mount point A directory on which a file system is to be mounted.

MTU See *maximum transmission unit*.

multicast A limited form of broadcast in which packets are sent to a subset of the possible hosts.

multihomed host A system that is attached to more than one network (has more than one physical network interface).

multipath Any device that can be reached through more than one component (e.g., more than one disk controller).

multiprocessor A system with more than one CPU.

multitasking The process of running more than one task at a time. Unless a system has more than one CPU, multitasking actually involves running tasks so that they take turns on the CPU and *appear* to be running at the same time.

multithreading The process by which a program can have more than one thread of control.

multiuser system Any system that can be used by more than one person at a time.

MX With respect to DNS, a mail exchange record. An MX record points to an SMTP mail server.

name A string associated with an entity (user, file, device, etc.) generally known by the system by a number or address.

named pipe A first in, first out (FIFO) file used for interprocess communication.

name space The collection of names stored in a particular system.

NAS Network-attached storage.

netgroup A collection of systems or users grouped together to facilitate their sharing of some network resource.

network The hardware, cabling, and other components and devices that connect systems together. The term generally refers to the equipment within organizational boundaries.

network-based file system A file system that is not stored and used on a single system. Contrast with *local file system*.

Network Information Service (NIS) A distributed network information service, originally called *Yellow Pages* (`yp`).

Network Information Service Plus (NIS+) The hierarchical and more secure network information service that was designed to replace NIS.

newfs A command for creating a new file system.

newsgroups Electronic message groups (e.g., `alt.doglovers`) distributed worldwide.

new-line character The character that marks the end of a line in Unix ASCII files, octal 12.

NFS The distributed network file system developer by Sun and available for almost every operating system.

niceness value The priority value given to a Unix process. It is called *niceness* because users can lower the priority of their own processes, allowing other processes to run more quickly.

NIS See *Network Information Service*.

NIS domain The collection of information available from a NIS server.

NIS maps Collections of network information available for use on a network.

NIS+ See *Network Information Service Plus*.

NIS+ tables Collections of network information available for use on a network.

node A general term, usually synonymous with *host*.

nonvolatile memory A memory device that does not lose its contents when power is lost.

nslookup A tool used to make DNS queries, usually used for debugging. Contrast with *dig*.

NVRAM Nonvolatile random-access memory.

OpenBoot Firmware for booting and running basic hardware diagnostics.

owner The person associated with a file or directory. Contrast with group.

package A Solaris-specific software format to facilitate installation and deinstallation.

packet Data packaged for transmission over a network, including data and host addresses.

paging The process of moving pages (chunks of data comprising processes and their data) in memory as needed to give all running processes opportunities to execute.

parameter An argument to a command or program.

parent process A process that has started another (child) process.

parity A simple method used to ensure that data is not corrupted during transmission.

parity error An indication that a transmission error has occurred.

partition A portion of the space on a disk, usually set up by the `format` program. Also called a *slice*.

password aging The process by which passwords eventually expire and must be replaced.

patch Generally, a replacement executable that repairs a bug in the original. Less frequently, a file insert for the same purpose.

patching Applying patches.

pathname The location of a file.

PC file system (pcfs) The file system type used to read and write MS-DOS-compatible floppies.

peripheral An external device (e.g., a printer or modem).

Perl The Practical Extraction and Report Language (or, as some prefer, the Pathologically Eclectic Rubbish Lister). Perl is an interpreted programming language that is especially adept at handling regular expressions and has many features missing from the standard shells.

permissions The set of bits that determines who can access a file and in what way.

ping A small ICMP packet generally used to test connectivity. `ping` is an *echo request* command. If the receiving system responds, the message `<hostname> is alive` is printed. Some say the name `ping` stands for Packet Internet Groper, but this is a woeful description of what the command does. Actually, the command was named because of its similarity to the sonar echo.

pipe A Unix command element that makes the output of one process the input of the next.

plex A group of one or more subdisks used in volume management.

Point-to-Point protocol (PPP) A protocol for running TCP/IP on a serial line. A successor to SLIP, it provides network (IP) connections over dial-up lines. Contrast with *SLIP*.

poll To periodically request status information from a device.

port A communications end point used by network services.

portability The feature of software that allows it to be moved and recompiled on systems different from the one it was developed on.

port mapper A network service that keeps track of communications channels.

port monitor A program that continuously watches for requests.

port numbers The numeric port identifiers used by TCP/IP. Most are stored in the file /etc/services.

postmaster The person who manages email. The alias that directs messages about bounced mail and such to this person.

power-on self test (POST) A set of tests run at boot time to check the hardware.

primary server With respect to DNS, a server that is authoritative and whose information is locally configured.

print client A system that uses a print server to print.

print queue A directory used to hold files waiting to be printed.

print server A system that queues print requests and sends them to the printer when the printer is ready.

private key In an asymmetric encryption system, the key that is known only to the owner of the key pair. Contrast with public key.

private network A network used for a special function, such as passing a heartbeat between systems so that they can monitor each other's availability.

process A program or command is called a process while it is running or waiting for resources so that it can be run.

process file system (procfs) A file system type used to provide interfaces to running processes and kernel structures.

process ID The numeric identifier of a running process.

process status The state of a process. Includes: running, stopped, and waiting, and so on.

procfs See *process file system*.

profile With respect to JumpStart, profile is a description of how a system is to be installed.

profile server With respect to JumpStart, a server that houses client profiles.

PROM Programmable read-only memory.

protocol A formal set of exchanges, and the rules that govern them, allowing computers to work together.

proxy A system that fronts for another in one way or another.

pseudo-file-system A file system that does not correspond to files on a disk, but to some other structure or interface (e.g., swap space or running processes).

pseudouser A username that is not associated with an actual individual, but with a function or process.

public key In an asymmetric encryption system, the key that is publicly available. Contrast with *private key*.

query A request for information on the part of a client system.

queue A holding space for requests (e.g., print requests).

quota A limit placed on users with respect to how much disk space they are allowed to use. For each user, there is a soft and hard limit determining when the user is warned about space and when the user can no longer create or edit files.

RAID Redundant array of inexpensive disks. RAID systems operate as a single large drive and, usually, incorporate some level of file redundancy—for example, mirroring or striping with parity—to ensure that file systems survive the loss of a single disk.

RAM Random-access memory. Sometimes referred to as *core* because early memory looked like tiny donuts with wires running through their cores. Contrast with ROM.

RARP See *Reverse Address Resolution Protocol*.

readme file A file containing information that the user should read before configuring or installing software.

recursive A computer program or procedure that calls itself.

redirection The process by which the output of a command or program is directed into a file, diverting it from its normal display.

refresh With respect to DNS, the time interval (in seconds) within which the secondaries should contact the primaries to compare the serial number in the SOA.

regular expression A pattern that selects a group of strings that have some component or substring in common. For example, the regular expression `\d` matches a single digit in Perl.

relative distinguished name A naming component used with LDAP that defines an object relative to its position in the tree.

relative pathname A pathname that is incomplete, relative to a particular directory (e.g., `bin/myproc`).

reliability The characteristic of a system in which it produces consistent accurate answers.

remote procedure call (RPC) A method for building client/server software.

remote shell (rsh) A command that allows the user to run commands on a remote system.

remote system A system other than the one the user is sitting at.

replica server A server which acts as a duplicate copy of another, to improve reliability and/or performance.

resolver With respect to DNS, a client that seeks information using DNS.

resource group The grouping of services such that they can be managed as a single entity.

resynchronization The process by which a volume manager rebuilds a drive after a failure.

Reverse Address Resolution Procotol (RARP) The protocol on TCP/IP networks used to map physical to IP (network) addresses.

rlogin A Telnet-like service that uses `~/.rhosts` files and the file `/etc/hosts.equiv` in the authentication process.

Rock Ridge file system Extension of the High Sierra format that allows contents of a CD-ROM to look and act like a Unix file system.

root file system The file system that contains files that describe the system.

root name servers With respect to DNS, the servers responsible for the root-level domain (.).

root user The superuser on Unix systems. Root is the privileged account and can generally do anything.

router A system that makes routing decisions.

routing The process of determining how to get packets from source to destination on a network or the Internet and getting them there.

run level The state in which a system is running, corresponding to the running processes and determined by the contents of the `/etc/rc?.d` directories. Also called *run state*.

run states The processing states of Unix systems, such as single-user and multiuser, that control how the system may be used and what processes are running.

runnable process A process that is ready to run. In other words, it is not waiting for any resources other than the CPU.

SAC See *Service Access Control*.

SAN Storage area network.

scalability The ability of a service to make efficient use of expanded resources.

screen lock A utility that locks a screen, requiring entry of the locking user's password to unlock it.

screensaver A utility that keeps a moving image on a screen or darkens it to prevent image from being burned into the display.

secondary server With respect to DNS, an authoritative server that obtains its information from a primary, using zone transfers.

Secure Shell (SSH) The de facto standard for encrypted terminal connections. Replaces Telnet in security-conscious organizations.

Secure Sockets Layer (SSL) A protocol providing an encrypted and authenticated communications stream.

seek A disk read operation that positions the read head at a certain location within a file.

segmentation fault An error that occurs when a process tries to access some portion of memory that is restricted or does not exist.

sendmail The primary mail exchange program on Unix systems.

serial number With respect to DNS, a special (magic) number that is used strictly to determine whether secondaries are up to date.

server A system that supplies services to other systems. A server can be a file server, print server, mail server, boot server, and so forth. A server can be a machine or a process.

Service Access Control (SAC) The Service Access Facility (SAF) master program.

shadow file The file that holds the encrypted passwords in Solaris and other SVR4-based Unix systems.

sharing Making file systems resources available to other clients on the network. See *exporting*.

shell A command interpreter that interacts with a user and passes commands on to the kernel.

shell script A file with shell commands that is made executable.

shell variable A variable that determines how a shell operates.

signal Any of a number of codes that can be sent to a process to tell it to do something. Typical signals tell a process to shut down (SIGKILL) or to reread its configuration file (SIGHUP). Signals are listed in the header file `/usr/include/sys/signal.h`.

Simple Network Management Protocol (SNMP) The most popularly used network management protocols.

slave server A system that answers queries for the service in question (e.g., NIS), but on which changes are *not* made. Contrast with *master server* and *replica server*.

slice Another word for *partition*.

Small Computer Systems Interface (SCSI) An industry standard bus used for connecting devices (e.g., disks and CD-ROMs) to a system.

SMTP Simple Mail Transfer Protocol.

SNMP See *Simple Network Management Protocol*.

SOA See *start of authority*.

socket A communications end point used by network communications—in particular, client/server applications.

sockfs A file system type that provides access to network sockets.

soft limit The lower limit, in disk quotas, at which the user is warned about the use of disk space, but still has time to react. Contrast with *hard limit*.

soft partitioning Resizable partitioning made possible through volume management.

software distribution Software as delivered by the manufacturer.

solution A grossly overused word describing hardware or software that answers a particular need.

source code Programming commands in a language (e.g., C) that requires compilation.

source file A file containing source code.

SPARC Scalable Processor Architecture from Sun. SPARC is a reduced instruction set (RISC) processor.

spawn A method by which a process creates another instance of itself.

specfs A file system that provides access to device drivers. Contrast with *fork*.

split brain The partitioning of a cluster due to a failure.

spooling Use of a special directory to hold files awaiting printing or some other processing.

SSH See *Secure Shell*.

SSL See *Secure Sockets Layer*.

stale NFS file handle A file handle that is out of date with respect to the corresponding file on the server.

standalone A system that does not require support from a server to run. Generally, standalone systems *are* connected to networks; the term is somewhat misleading.

standard error The device to which Unix commands send error messages—by default, the monitor or terminal.

standard input The device from which Unix commands receive input—by default, the keyboard.

standard output The device to which Unix commands send output—by default, the display or terminal.

start of authority (SOA) With respect to DNS, a record that contains the information that other name servers querying this one will require, and will determine how the data is handled by those name servers.

state database A storage area in which a volume manager keeps track of volumes.

static RAM (SRAM) A RAM device that holds its contents as long as it receives power.

static distributed database A database that is distributed across numerous systems or sites but remains relatively stable.

stealth server An authoritative server that is not listed.

stopped job A process that has been halted but can be restarted.

striping Combining multiple disks into a single logical disk in such a way that the data is “striped” across all drives.

stub zone A zone that only replicates the NS records of a master zone.

subdirectory A directory located within another directory. Any directory, other than `/`, can be called a subdirectory.

subdisk A portion of a disk under volume management.

subdomain With respect to DNS, a portion of a domain for which an authoritative name server exists.

subnet mask A mask that breaks a network address space into separate subnets.

subnetwork (subnet) A portion of a network separated off, generally by a router, to reduce traffic.

superblock A block on a disk that contains information about file systems.

SuperSPARC Module A card containing the SuperSPARC processor, memory, and cache controller.

superuser The privileged root user.

suspend To halt a process temporarily.

swap file A file used in addition to the swap partition.

SWAPFS A pseudo-file-system used for swapping.

swapping The process by which, when the demand for memory is excessive and the system is having difficulty maintaining the free memory list, entire processes (rather than just pages), are replaced in memory. Contrast with *paging*.

swap space The memory/disk area used to transfer programs between disk and memory.

symbolic link A special file that points to a file or directory. The contents of a symbolic link is the pathname to the file pointed to.

symmetric encryption An encryption scheme that uses the same key for encryption and decryption.

symmetric multiprocessing A form of multiprocessing in which multiple processors can run kernel-level code and run with equivalent status.

synchronous Under the control of a timing signal.

syslog A general-purpose logging service available in Unix systems. `syslog` maintains logs on behalf of many services. Its configuration file is `/etc/syslog.conf`.

system administrator The person responsible for configuring and managing systems, especially servers.

system board On Sun systems, a circuit board with CPUs installed.

system call A request by a program for kernel support.

tar file A file, usually for archiving, created by the `tar` command.

TCP See *Transport Control Protocol*.

TCP/IP See *Transport Control Protocol/Internet Protocol*.

Telnet The virtual terminal tool used most frequently to login to remote systems. Contrast with `rlogin` and `ssh`.

temporary file system (TMPFS) A file system that uses memory and swap space. See *swapping*.

terminal A physical device or pseudo-device for logging in and interacting with a computer.

terminal type The name that identifies a physical terminal or emulation.

TFTP See *Trivial File Transfer Protocol*.

third party A general term used to describe a product that is not associated with your system's manufacturer or your own development efforts. That is, if it's not you and it's not Sun, it's third party.

time to live (TTL) With respect to DNS, the length of time that an address is valid before an authoritative name server needs to be asked for it again.

TMPFS See *temporary file system*.

TODC Time-of-day clock.

token (1) An indivisible unit of a programming language; (2) A succession of bits sent from one system to another to pass control (as in a token ring network); (3) A hand-held device used in certain three-factor security systems (e.g., SecurID).

top A utility for looking at processes—especially useful in determining which processes are using most of the CPU resources.

traceroute A tool that traces the route that a packet takes over the Internet. Uses TTL and ICMP time-exceeded messages to report successive nodes in the route and timing.

traffic The collection of packets (frames) on a network.

Transport Control Protocol (TCP) The major transport protocol in TCP/IP. TCP provides a reliable connection-oriented stream delivery service and is one of the core Internet protocols. TCP provides a full-duplex connection between two machines on an internet and provides for efficient data transfer across networks across different communications protocols.

Transport Control Protocol/Internet Protocol (TCP/IP) A suite of protocols that forms the primary protocol on the Internet and most heterogeneous networks.

Transport Layer Security (TLS) A protocol (and the IETF group working on it) for encrypted and authenticated communications. TLS is based on SSL.

tree With respect to DNS, the entire structure starting from the root and descending to each and every host.

Trivial File Transfer Protocol (TFTP) A simple file transfer tool.

trust The process by which one system extends privileges to another (e.g., accepting a remote system's authentication of a user). On Unix systems, trust involves the `/etc/hosts.equiv` and `~/.rhosts` files.

tunneling The process by which one network connection is used to host others.

UDP See *User Datagram Protocol*.

UFS UNIX File System.

UID User ID.

UID number The numeric ID associated with a user.

unbundled Not included as part of the original purpose. For Solaris, not part of the operating system, but purchased separately.

Unix An operating system descending from the original UNIX developed at Bell Labs in 1969. The key features of Unix are that it is an interactive, multiuser, time-sharing system with a very modular design. Few of us today can remember who exactly it is that owns the term or whether the name, in fact, can be validly used when talking about Solaris or Linux. At the same time, we call ourselves Unix system administrators and everyone knows exactly what we mean.

unmount The process of removing a mount; removing access to a remote file system.

User Datagram Protocol (UDP) Part of the TCP/IP protocol suite. A connectionless protocol above IP in the protocol stack, it includes a protocol port number for the source and the target, allowing it to distinguish between different application programs on the source and target systems.

username The string of characters used to identify a user. The username is associated with the UID in the `passwd` file.

virtual memory The extension of physical memory to a larger space.

volatile memory Memory that loses its data when there is no power. Compare with *nonvolatile memory*.

volume A virtual disk; a space that acts as if it were a physical disk provided through volume management.

Web Flash A Sun technology for installing a system and creating an archive used to clone the system.

Web Start Flash archive A snapshot of a system that has been set up and configured for cloning.

wildcard A character, or metacharacter, that matches one or a number of characters. For example, `?` in a shell represents a single character.

wrappers A security process that involves adding a layer of security to a common service.

X See *X11* and *X Windows System*.

X11 The *X Window System*, Version 11, developed at MIT.

X Display Manager (XDM) An OpenWindows program that manages X displays.

xdmcp X display management console protocol.

X server In the X protocol, the server is the system that serves windows to a user (i.e., the user's local system).

X Window System A network-based (as opposed to local) windowing system developd at MIT.

yp An early name for NIS. The acronym stands for "Yellow Pages," but this name is no longer used because it is a trademark of British Telecom.

zone With respect to DNS, a boundary for a name server's authoritativeness. A zone relates to the way the DNS database is partitioned and distributed.

Bibliography and Recommended Reading

Bialaski, Tom. "Implementing LDAP in the Solaris Operating Environment." *Sun Blueprints OnLine*, October 2000.

Bialaski, Tom. "Exploring the iPlanet Directory Server NIX Extensions." *Sun Blueprints OnLine*, August 2000.

Bialaski, Tom. "NIS to LDAP Transition: Exploring." *Sun Blueprints OnLine*, February 2000.

Birdsall, James W., *Sun Hardware Reference*. http://www.si.unix-ag.org/faqs/SUN-HW-FAQ.html, 1994.

Budlong, Mo. "Command line Psychology 101." *SunWorld*, February 1999.

Budlong, Mo. "Getting started with Perl, Part 1: An introduction to the language that 'can do anything.'" *SunWorld*, May 1999.

Budlong, Mo. "Getting started with Perl, Part 2: Tricks for calling Perl functions by reference." *SunWorld*, June 1999.

Budlong, Mo. "Tips on good shell programming practices: What #! really does." *SunWorld*, September 1999.

Cockcroft, Adrian. "Adrian Cockroft's frequently asked (performance) questions." *SunWorld*, ongoing.

Cockcroft, Adrian. "Disk error detection: What tools can warn you about disk failure automatically—and efficiently—before your disks fail?" *SunWorld*, July 1999.

Cockcroft, Adrian. "How to optimize caching file accesses." *SunWorld*, March 1997.

Cockcroft, Adrian. "Performance Q&A compendium." *SunWorld*, December 1995.

Cockcroft, Adrian. "Prying into processes and workloads." *SunWorld*, April 1998.

Cockcroft, Adrian. *Sun Performance and Tuning: Java and the Internet*, Second Edition. Santa Clara, CA: Sun Microsystems, 1998.

Cockcroft, Adrian. "Upgrades for SyMON and SE: What's changed in these helpful utilities? Find out here." *SunWorld*, February 1999.

Cockcroft, Adrian. "What does 100 percent busy mean?" *SunWorld*, August 1999.

Cockcroft, Adrian. "What's the best way to probe processes?" *SunWorld*, August 1996.

Cook, Rick. "Solaris and Windows NT: The odd couple gets cozy." *SunWorld*, January 1999.

Coomer, James and Cahru, Chaubal. *Introduction to the Cluster Grid—Part 1*. Santa Clara, CA: August 2002.

Costales, Bryan, with Allman, Eric. *sendmail*, Second Edition. Sebastopol, CA: O'Reilly & Associates, 1997.

Galvin, Peter. "Enter the Secure Shell." *SunWorld*, February 1999.

Galvin, Peter. "More on mastering the Secure Shell." *SunWorld*, March 1999.

Galvin, Peter. "The power of /proc: Using proc tools to solve your system problems." *SunWorld*, April 1999.

Galvin, Peter. "Stop your fire-fighting." *SunWorld*, September 1999.

Henry, S. Lee. "Pitching Patches." *SunExpert*, August 1998.

Henry, S. Lee and Graham, John R. *Solaris 2.x: A System Administrator's Guide*. New York: McGraw-Hill, 1995.

Howard, John S. "An Introduction to Live Upgrade." *Sun BluePrints OnLine*. July 2000.

Howard, John S. *Using Live Upgrade 2.0 With JumpStart Technology and Web Start Flash*. Santa Clara, CA: Sun Microsystems, April 2002.

Howard, John S. and Noordergraaf, Alex. *WebStart Flash*. Palo Alto, CA: Sun Microsystems, November 2001.

Jones, Sharis L. "Which server is best when mixing Windows NT into a Solaris network?" *SunWorld*, August 1999.

Kasper, Paul Anthony and McClellan, Alan L. *Automating Solaris Installations: A Custom JumpStart Guide*. Englewood Cliffs, N.J.: Sun Microsystems/Prentice Hall, 1996.

Laird, Cameron and Soraiz, Kathryn. "POP goes the server: What considerations should you take into account when choosing a POP service for your Unix e-mail server?" *SunWorld*, May 1999.

Marks, Evan. "Create a highly available environment for your mission-critical applications." *SunWorld*, August 1997.

Mauro, Jim. "Processor utilization and wait I/O—a look inside." *SunWorld*, August 1997.

McDougall, Richard. "Getting to know the Solaris filesystem, Part 1: Learn all about the inner workings of your on-disk filesystem, including allocation management, storage capacity and access control list support, and metadata logging. *SunWorld*, May 1999.

McDougall, Richard. "Getting to know the Solaris filesystem, Part 2: What factors influence filesystem performance?" *SunWorld*, May 1999.

McDougall, Richard. "Getting to know the Solaris filesystem, Part 3: How is file data cached? Interactions between the filesystem cache and the virtual memory system explained. *SunWorld*, July 1999.

Musciano, Chuck. "How to manage and implement change in your Unix environment: When are changes necessary—and when should they be avoided?" *SunWorld*, March 1999.

Musciano, Chuck. "RAID basics, Part 3: Understanding the implementation of hardware- and software-based solutions: What are the benefits and drawbacks of each?" *SunWorld*, August 1999.

Musciano, Chuck. "RAID basics, Part 4: Uniting systems and storage: Learn how to connect your new RAID system to your servers." *SunWorld*, September 1999.

Pérez, Juan Carlos. "Samba Windows-Unix tool is updated — Web-based admin capabilities added." IDG News Service, January 15, 1999.

Ramsey, Rick. *All About Administering NIS+, Second Edition.* Englewood Cliffs, N.J.: Sun Microsystems, 1996.

Shah, Rawn. "Building a reliable NT server, Part 1." *SunWorld*, January 1999.

Shah, Rawn. "Building a reliable NT server, Part 2." *SunWorld*, February 1999.

Shah, Rawn. "Building a reliable NT server, Part 3." *SunWorld*, March 1999.

Shah, Rawn. "Building a reliable NT server, Part 4." *SunWorld*, April 1999.

Shah, Rawn. "Building a reliable NT server, Part 5." *SunWorld*, May 1999.

Shah, Rawn. "Microsoft steps further into NT-Unix integration." *SunWorld*, December 1998.

Shah, Rawn. "Storage beyond RAID." *SunWorld*, July 1999.

Shah, Rawn. "What is RAID? And what can it do for your network?" *SunWorld*, June 1999.

Snevely, Rob. *Solaris 8 Operating Environment Additions to sysidcfg*. Palo Alto, CA: Sun Microsystems, March 2000.

Stern, Hal. "ARP networking tricks." *SunWorld*, May 1997.

Stern, Hal. "An automounter and NFS potpourri." *SunWorld*, July 1996.

Stern, Hal. "Here's how you too can understand device numbers and mapping in Solaris." *SunWorld*, December 1996.

Stern, Hal. "How can you make the network yente work for you?" *SunWorld*, March 1997.

Stern, Hal, Eisler, Mike, and Labiaga, Ricardo. *Managing NFS and NIS*, Second Edition. Sebastopol, CA: O'Reilly & Associates, 2001.

Stern, Hal. "Twisty little passages all autmounted alike." *SunWorld*, June 1996.

Stern, Hal. "The Unix Automounter." *SunWorld*, May 1996.

Sun Microsystems. *LDAP Setup and Configuration Guide*. Palo Alto, CA, January 2001.

Sun Microsystems. *NIS+ to LDAP Migration in the Solaris 9 Operating Environment: A Technical White Paper*. Palo Alto, CA, 2002.

Sun Microsystems. *Administrator's Guide: iPlanet Directory Service, Version 5.1*. Palo Alto, CA, February 2002.

Sun Microsystems. *Solaris 9 Resource Manager Software: A Technical White Paper*. Palo Alto, CA, 2002.

Sun Microsystems. *Sun Cluster 3.0 System Administration Guide*. Palo Alto, CA. November 2000.

Sun Microsystems. *System Administration Guide: Advanced Administration*. Santa Clara, CA. May 2002.

Sun Microsystems. *Solaris WBEM Services Administration Guide*. Santa Clara, CA. May 2002.

Sutton, Steve. *An Intro to Windows NT Security*. http://www.trustedsystems.com, October 24, 1997.

Veritas. *Veritas Volume Manager 3.2 Administrator's Guide: Solaris*. Mountain View, CA. July 2001.

Wong, Brian L. *Configuration and Capacity Planning for Solaris Servers*. Upper Saddle River, N.J.: Sun Microsystems/Prentice Hall, 1997.

Zajac, Blair. "Watching your systems in realtime: What tools can you use and how do they work?" *SunWorld*, July 1999.

Special credits: Much of the material in Chapter 13 was first presented as a column in *SunWorld* magazine, August 1997. Some of the material in Chapter 5 was first presented as a column in *SunExpert*, August 1998.

Index