ISO 9001:2000 Certified

Treklogic Inc
2 Sheppard Ave. E. Suite 700
Toronto, ON M2N 5Y7, Canada
Telephone 416.225.9336
Fax 416.225.7991

Linux to Solaris Administrators Guide

Author: Dan Sinclair, Edsel Adap and Frederick Sing-yan Tam
Version: 1.0
Date: Jan 08, 2007

Contents

Preface **ix**

1 Overview of Linux and Solaris Differences **1**

Architectural Similarities and Differences 2

File System Organization 2

Locations of Common Commands 3

Location of Configuration Files 4

Networking 4

File System 4

Mail 5

Location of Log Files 5

Script Migration 6

2 Solaris Features not in Linux **7**

SMF 8

Zones 11

DTrace 14

ZFS 16

Predictive Self Healing 18

3 Command Differences **19**

awk 19

basename 19

cat 20

chown . . . 20
df . . . 21
du . . . 22
ps . . . 22
setfacl . . . 23
getfacl . . . 25
tar . . . 25
useradd . . . 25
groupadd . . . 27

4 Installation **29**
CD/DVD Media Installation . . . 30
Network Installation . . . 30
Flash Archives . . . 32
Live Upgrade . . . 32

5 Software Management **33**
RPM packages . . . 34
Solaris Packaging . . . 34
Patching . . . 34

6 System Management **35**
Booting, Shutdown and Run Levels . . . 36
System Services . . . 37
User/Group Management . . . 38
Printing and Printer Management . . . 38
File System Management . . . 39
Loopback Devices . . . 40
File System Quotas . . . 40
Disk and Volume Management . . . 40
Disk Management . . . 40
Volume Management . . . 41

Network Management 41
Remote Management 43
Kernel Configuration 43
Loading Kernel Modules 43
Kernel tuning Commands 44

7 Device Management **45**
Device Naming and Access 46
Adding/Removing Devices 46
Removable Media 46
Tape Drives 47
Terminals, Modems and Serial Ports 47

8 Security and Hardening **49**
Hardening Tools 50
Solaris Security Toolkit 50
RBAC 50
BART 50
Least Privileges 50
Auditing Tools 51
Securing and Removing Services 51
Kernel Tuning for Security 51

9 Monitoring and Performance **53**
Processors 54
Memory 54
Network 55
Disks, Volumes and File Systems 56
System and User Processes 57
Input/Output 57

10 Backup and Restore **59**
File System Backup and Restore 60
Compression Tools 60
File System Snapshots 60

11 Troubleshooting **61**

Installation . . . 62

Installing from a USB CDROM drive . . . 62

System Startup . . . 62

Cannot mount boot archive . . . 62

Core Files . . . 63

Kernel Crash Dumps . . . 63

Logs . . . 63

Missing Commands . . . 64

Printing . . . 64

File Systems . . . 64

Root Password Recovery . . . 64

Network . . . 64

Controlling NFS versions supported . . . 65

Diagnostic and Debugging Tools . . . 65

A Packages **67**

Network Servers & Clients . . . 67

Commands . . . 67

Libraries . . . 68

Compilers & Tools . . . 68

Scripting Languages . . . 68

Security Tools . . . 68

Shells . . . 68

Applications . . . 68

Networking . . . 68

Publishing . . . 68

Utilities . . . 69

Accessibility . . . 69

Editors . . . 69

Development . . . 69

Tools . . . 69

Languages 69
Libraries 70
Desktop 70
Environment 70
System 70
Daemons 70
X 70
Applications 70
Window Managers 70

B Quick Reference Guide **71**

List of Tables

1.1 Common file system directories 3
1.2 Common command directories 3
1.3 Extra Solaris command directories 3
1.4 Solaris commands in /usr/ucb 4
1.5 Linux and Solaris networking configuration files 5
1.6 Linux and Solaris file system configuration files 5
1.7 Linux and Solaris mail configuration files 5

2.1 SMF Commands 9

3.1 Solaris awk variations 20
3.2 Linux vs Solaris cat arguments 20
3.3 Additional Solaris /usr/bin/chown arguments 21
3.4 Additional Linux chown arguments 21
3.5 Linux and Solaris du comparison 22
3.6 Linux du options without Solaris equivalents 23
3.7 Linux setfacl options not available in Solaris 24
3.8 Linux and Solaris useradd differences 26
3.9 Linux and Solaris groupadd differences 27

6.1 Linux run levels 36
6.2 Solaris run levels before Solaris 10 36
6.3 Solaris 10 milestones 36
6.4 Solaris 10 run levels 37

6.5 Linux services configuration locations 37
6.6 Solaris services configuration locations 38
6.7 System V print commands 38
6.8 File System Types 39
6.9 Metadevice commands 41
6.10 Networking tools 42
6.11 Networking configuration files 42

11.1 Core file utilities 63
11.2 Useful networking commands 65

B.1 Command Differences 71
B.2 Configuration Files 71
B.3 Kernel Drivers 72
B.4 Kernel Configuration 72

Preface

The aim of the *Linux to Solaris Administrator Guide* is to give Linux administrators the information and guidance they'll need to make a successful transition to using Solaris 10.

The Linux to Solaris Administrators Guide is not intended for a first time system administrator. The guide assumes a certain amount of background administering a Linux system.

The major topics in this guide include:

Chapter	*Description*
Overview	Overview of Solaris and Linux differences.
Solaris Features not Available in Linux	An overview of some of the new features in Solaris 10 that either don't exist, or are still in the development phase, on Linux.
Command Differences	Information on command differences between Linux and Solaris.
Installation	Information on installation and its comparison to Linux.
Software Management	Information on software management differences in Solaris as compared to Linux.
System Management	Information on system management differences in Solaris as compared to Linux.
Device Management	Information on device management differences in Solaris as compared to Linux.
Security and Hardening	Information on security and system hardening differences in Solaris as compared to Linux.
Monitoring Performance	Information on system monitoring and performance in Solaris as compared to Linux.
Backup and Restore	An overview and comparison of backup and restore procedures between Solaris and Linux.
Troubleshooting	Information on troubleshooting common Solaris issues and errors.

After reviewing the material in this book you should have enough exposure to be able to install and maintain a Solaris machine.

Chapter 1

Overview of Linux and Solaris Differences

While similar in many respects there are still quite a few differences between a Linux and Solaris machine including commands, file systems and heritage. These differences can be as simple as a renamed configuration file to the more complicated alternate meanings for command line arguments.

This chapter presents an overview of some of the differences between Linux and Solaris.

Topics Covered
Architectural Similarities and Differences
File System Organization
Locations of Common used Commands
Location of Configuration Files
Location of Log Files
Script Migration

Architectural Similarities and Differences

While similar in many respects there are fundamental differences between the Solaris and Linux operating environments.

Throughout the development of Solaris the focus has been on compatibility. Compatibility with previous releases (binaries compiled on Solaris 8 will run on Solaris 10) and compatibility with de jure standards such as POSIX. Linux has had the freedom to establish new de facto standards (such as extended command line syntax, with a strong consistency among commands). However, judicious use of the included GNU utilities, the proper command line settings, downloads of additional utilities, and some acquired knowledge such as this guide and sites such as the Rosetta Stone[1] will mitigate most differences.

Many of the system similarities can be attributed to the implementation of different system standards. These standards make it a lot easier to move applications between Operating Systems. Solaris conforms to the POSIX, SVID and XPG standards.

Although many commands may have the same name, the implementation and command line options may have changed. The man pages should be consulted to verify the functionality of commands.

The first difference you'll probably notice is the default shell. The default Linux shell is Bash. Linux distributions typically symlink */bin/sh* to */bin/bash*. In Solaris, on the other hand, */bin/sh* remains the Bourne shell; this is done to preserve absolute compatibility with existing shell scripts. Bash is upwards compatible from the Bourne shell, but includes many additional extensions; Bash is available on Solaris as */bin/bash*. You may want to make a root account with */bin/bash* as its shell.

Depending on the kernel configuration a Linux kernel will be either dynamically linked or monolithic. The Solaris kernel is *always* dynamically linked.

The standard Solaris file system format is *UFS*; the extra-cost file systems VxFS and QFS are often used as well. Beginning with Solaris 10 6/06 Solaris ZFS is also available. Linux will typically use one of *ext3*, *reiser*, *JFS* or *XFS*.

File System Organization

There are many similarities between Linux and Solaris file system hierarchies. That being said, there are slight variations in the usage of some directories.

Table 1.1 lists the common file system directories.

Of these directories only one directory has differences worth noting, */proc*. On Linux, */proc* contains information on the current system configuration and

[1] http://bhami.com/rosetta.html

Table 1.1: Common file system directories

/	/sbin	/bin
/lib	/usr	/etc
/var	/opt	/proc

process along with files you can alter to update kernel variables and process information. On Solaris the */proc* directory only contains process information. You won't use */proc* to update kernel tunables or retrieve system information.

Solaris adds the */platform* directory which contains platform specific information and applications. There is no equivalent to */platform* on Linux.

For a full description of the file system organization on Solaris please refer to the *filesystem(5)* manpage.

Locations of Common Commands

One of the more frustrating aspects of changing operating environments is determining the location of commonly used commands. There is a common set of command directories between Linux and Solaris which are listed in Table 1.2.

Table 1.2: Common command directories

/bin	/usr/bin
/sbin	/usr/sbin

In order to maintain compatibility with System V, BSD and GNU software Solaris also includes several extra command directories. These extra directories can be seen in Table 1.3.

Table 1.3: Extra Solaris command directories

/usr/openwin/bin	/usr/dt/bin	/usr/sfw/bin
/opt/sfw/bin	/usr/xpg4/bin	/usr/ccs/bin
/usr/ucb		

/usr/bin contains the standard System V implementation of certain applications. */usr/ucb* contains implementations that are compatible with legacy BSD versions. Some commands exist in both */usr/bin* and */usr/ucb* with differing

implementations. You'll need to determine which version meets your requirements when doing script migrations and setup your PATH as required. Table 1.4 lists some */usr/ucb* commands which may differ from the implementation of the same command in */usr/bin*.

Table 1.4: Solaris commands in /usr/ucb

basename	df	du	echo	expr	fastboot
fasthalt	file	from	groups	install	ld
lint	ln	lpc	lpq	lpr	lprm
lptest	ls	mkstr	printenv	ps	rusage
sed	shutdown	stty	sum	test	touch
tr	tset	users	vipw	whereis	whoami

Along with the System V and BSD versions there are two directories that contain free software. These are the */usr/sfw/bin* and */opt/sfw/bin* directories. */usr/sfw/bin* will contain any freeware installed off the Install media while */opt/sfw/bin* contains software installed off of the Companion CD.

The free software versions are the same implementations as found on a Linux machine. These GNU commands on Solaris are typically prefixed with a *g*. (e.g. tar vs gtar).

As Solaris matures, some software that used to be shipped with the Companion CD slowly makes its way into the Solaris install media. You may notice that on a given release of Solaris a command in */opt/sfw/bin* may migrate to */usr/sfw/bin*.

Note that software delivered in */usr/sfw* is fully supported by Sun through Sun's standard support channels. Software delivered in */opt/sfw* is community supported. These are simply packaged by Sun for convenience. If support issues arise regarding software in */opt/sfw*, the usual open source support channels must be used.

Location of Configuration Files

Networking

Table 1.5 lists the Linux and equivalent Solaris networking configuration files.

File System

Table 1.6 lists the Linux and equivalent Solaris file system configuration files.

Table 1.5: Linux and Solaris networking configuration files

Linux	*Solaris*
/etc/ntp.conf	/etc/inet/ntp.conf
/etc/inetd.conf	/etc/inet/inetd.conf
/etc/sysconfig/network-scripts/ifcfg-{interface}	/etc/inet/netmasks
/etc/networks	/etc/inet/networks

Table 1.6: Linux and Solaris file system configuration files

Linux	*Solaris*
/etc/fstab	/etc/vfstab
/etc/exports	/etc/dfs/dfstab (format is different)
/etc/auto.master	/etc/auto_master
/etc/auto.home	/etc/auto_home

Mail

Table 1.7 lists the Linux and equivalent Solaris mail configuration files.

Table 1.7: Linux and Solaris mail configuration files

Linux	*Solaris*
/etc/aliases	/etc/mail/aliases
/etc/mail.rc	/etc/mail/Mail.rc
	/etc/mail/mailx.rc

Sendmail configuration on Linux is typically stored in the /etc/mail directory. On versions of Solaris up to Solaris 9 this configuration was stored in /usr/lib/mail. With Solaris 10 the /usr/lib/mail directory is a symlink to the /etc/mail directory.

Location of Log Files

There is one main log file directory used on a Linux system, */var/log*. This is where the log files for the various system daemons are stored.

Solaris uses a slightly different setup for its log directories. */var/log* stores the syslog and authlog files and */var/adm* stores the messages log file, */var/log/messages*. This is the file that contains the logs for everything. (By default, this is configurable in the syslog.conf file.)

Script Migration

There are a few pieces of information that need to be gathered when porting shell scripts from Linux to Solaris.

First, make sure that any file system paths used by the scripts are valid on Solaris 10. As mentioned in Section 1 some of the commands maybe in different locations, or have different implementations. If there are GNU versions of the commands available, via the Software Companion CD, alter the path to utilize this version. (Check both /usr/sfw/bin and /opt/sfw/bin to see if the GNU version exists).

Once you've got all the command paths verified, check that the command line arguments are still correct on Solaris. As discussed in Chapter 3, Linux commands that utilize the long option names (–option-name) may be candidates for modification. The Solaris equivalents of those commands most likely don't understand the long options, unless you're using the GNU versions.

If you are taking any command output and parsing it to extract specific information you may need to re-write the parsing routines to match the output of the program on Solaris.

Chapter 2

Solaris Features not in Linux

With the release of Solaris 10, and the subsequent updates, there are several features available that provide a unique value proposition for the utilization of Solaris 10: features ranging from virtualization, to application tracing and service management. These tools make it easier to harness the full potential of your hardware and to track down any latent performance issues.

This chapter will provide a brief overview of four key features now available in Solaris 10: service management with the Solaris Service Manager (SMF), Virtualization using Zones, dynamic tracing using DTrace, self healing and the Solaris ZFS.

While all of these features are very powerful tools provided with Solaris 10, only SMF is required learning when transitioning to Solaris 10. All other features, Zones, DTrace and ZFS included, can be set aside and incorporated at your own pace.

Topics Covered
SMF
Zones
DTrace
ZFS
Predictive Self Healing

SMF

The Service Management Facility (SMF) is the new system to manage services in Solaris 10. SMF takes over the role that was previously delegated to scripts in the */etc/rc*.d* directories, although those scripts will continue to run.

Your first encounters with a Solaris 10 machine may leave you wondering how to manage any running services. The */etc/rc*.d/* directories are relatively empty compared to Linux. The reason for this is SMF. The majority of the system services available under Solaris 10 are controlled by SMF. In order to control these services you'll need to get familiar with a couple of SMF commands: *svcs(1)* and *svcadm(1M)*. These two commands will allow you to start, stop, diagnose and control the services previously controlled through */etc/rc*.d*.

Enabling and disabling of services is done with the *svcadm* command. As can be seen in Figure 2.1 we use the *enable* and *disable* subcommands to modify the service state.

Figure 2.1: Modifying service states

```
# svcadm enable network/http:apache2
# svcadm disable network/http:apache2
```

SMF service states are persistent. If you enable or disable a service and reboot your machine the service will be restored to the state last specified.

There are several other subcommands to *svcadm* that will be beneficial to know when working with a Solaris 10 machine. See *svcadm(1M)* for the full list of subcommands.

Information on the services running, or available, on the system can be obtained using the *svcs* command. Figure 2.2 shows some possible output from *svcs -a*.

There are a few things to note in Figure 2.2.

First notice how the services are named. Each service has a unique FMRI (fault management resource identifier) that identifies the service. Some of the services have an *instance* name on the end (the :default of svc:/network/ssh:default). There can be multiple instances of a single service executing on the system at one time.

Second, notice the *svc:/milestone/multi-user-server:default* FMRI. SMF uses a system of milestones instead of run levels to specify the system state. We can see that I currently have *milestone/multi-user-server* enabled. More information on Solaris 10 milestones can be found in the System Management chapter.

The final thing to note is the STATE column. This column is telling us the current state of each service. The *legacy_run* state is for services that are still

Figure 2.2: System services

```
# svcs -a
STATE        STIME    FMRI
legacy_run   15:33:43 lrc:/etc/rc3_d/S84appserv
legacy_run   15:33:44 lrc:/etc/rc3_d/S90samba
...
disabled     15:33:33 svc:/network/shell:kshell
disabled     15:33:33 svc:/network/talk:default
...
online       15:33:44 svc:/milestone/multi-user-server:default
online       15:33:47 svc:/system/zones:default
...
offline      15:33:17 svc:/network/ssh:default
offline      15:33:18 svc:/application/print/ipp-listener:default
```

executing from the old */etc/rc*.d* scripts. The *disabled* state specifies all of the services that have been disabled on the system. The *online* state lists all of the services that are currently enabled on the system. Lastly, *offline* lists all of the services that were unable to start, or had a service fault, and have been taken offline by SMF.

Typically any *offline* services will need to be investigated to determine the reason for their off-lining. The *svcs -x* command can be used to gather information on the off-lined processes. Figure 2.3 shows that the *svc:/network/physical:default* service is disabled. Adding the *-v* flag we can see that this is causing the *svc:/network/ssh:default* service to be off-lined.

The *svcs -x* output also gives pointers on where you can go for information on the current issue. These can be pointers to websites, log files or man pages. These are typically very handy resources in diagnosing and correcting the issue at hand.

Along with *svcs* and *svcadm* there are several other commands that are used to interact with SMF. These commands are listed in Table 2.1.

Table 2.1: SMF Commands

Command	*Description*
svccfg	Configure services.
inetadm	Administer inetd services.
inetconv	Convert inetd services to SMF.

The *inetd* services have been converted to run under SMF. Inetd still functions as it did previously, launching the required service as a request comes in,

Figure 2.3: SMF state explanations

```
# svcs -x
svc:/network/physical:default (physical network interfaces)
 State: disabled since Thu Sep 28 15:33:17 2006
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: ifconfig(1M)
Impact: 5 dependent services are not running.  (Use -v for list.)

# svcs -xv
svc:/network/physical:default (physical network interfaces)
 State: disabled since Thu Sep 28 15:33:17 2006
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: man -M /usr/share/man -s 1M ifconfig
Impact: 5 dependent services are not running:
        svc:/milestone/network:default
        svc:/network/nfs/nlockmgr:default
        svc:/network/nfs/client:default
        svc:/network/nfs/status:default
        svc:/network/ssh:default
```

it's just controlled through the unified SMF interface.

When a service is defined in SMF you can also define any dependencies of that service. For example, PostgreSQL depends on a local file system. With that knowledge, SMF knows that if the user disables the file system that PostgreSQL depends on it also needs to disable PostgreSQL.

This dependency information is also used by SMF to execute as much of the boot processes in parallel as possible. SMF can take any services which don't depend on each other and start them at the same time. This can result in quicker boot times as compared to the linear processing of the */etc/rc*.d* scripts.

There are three main types of services provided by SMF. *Transient*, *Wait* and *Contract* services. Transient services are often configuration services requiring no long-running process. Wait services run for the lifetime of the child process and are restarted when the process exits. Contract services are the standard system daemons and require processes which run forever once started. The death of all processes in a contract service is considered a service error which will cause the service to restart.

More information on SMF can be found in the *Solaris Service Management Facility - Quickstart Guide* [1] and the applicable sections of the *Solaris Administration Guide: Basic Administration*[2].

Zones

Virtualization is quickly becoming a hot topic in system administration. The ability to run multiple instances of an operating system on a single machine allows us to increase the overall usefulness of that machine. We're able to take some of the idle CPU time, free memory and unused disk and run two, three or four parts of our network architecture.

Sun has taken this technology in hand with Solaris 10 and presents the concept of Zones. (You may also see references to Solaris Containers, which are Zones coupled with Resource Management.)

If you've heard of chroot jails, originating from FreeBSD, you'll understand the basic concept of Zones. When utilizing Zones there is one instance of the Solaris kernel. Even though there is only one kernel, all of the zones, except in one special case, are segregated and isolated from each other. (The special case being the *global zone* which can see everything.)

Each zone on the system can have its own IP address, resource controls, root and user accounts, running services and installed software. The only way for two zones to communicate is using the TCP/IP networking capabilities.

[1] http://www.sun.com/bigadmin/content/selfheal/smf-quickstart.html
[2] http://docs.sun.com/app/docs/doc/817-1985

There has been a lot of work to make the creation and interaction with zones as simple as possible. There are two main commands to remember, *zonecfg* and *zoneadm*. With these two tools you'll be able to create, install, boot, halt, uninstall and delete your zones.

zonecfg and *zoneadm* provide a simple, yet powerful, interface to work with Zones. We can get a list of the current Zones using the *list* subcommand to *zoneadm*. An example of this can be seen in Figure 2.4. There are a couple of things to note. First, there is always a global zone. The global zone is the main machine and can see everything that happens in all other zones. You'll also notice there is no ID assigned to the test zone. The Zone IDs are only assigned after a Zone has been booted. These IDs can change depending on the order of zone startup. All of the zone work we do will use the zone name which must be unique.

Figure 2.4: Listing zones

```
# zoneadm list -cv
ID NAME             STATUS         PATH
 0 global           running        /
 - test             configured     /zones/test
```

There are two types of zones, *whole* and *sparse* root Zones: the difference between the two being the amount of data that is copied into the zone file system. With a *whole root* zone all of the needed data is copied from the global zone. With a *sparse root* zone the */usr*, */sbin*, */platform* and */lib* directories are read-only loopback links to the global zone. Using a *sparse root* zone provides a lot of disk savings over a *whole root* zone but with the drawback that you can't install anything into the loopbacked file systems from within the local Zone.

The first step in creating a zone is to create its configuration. This is done with the *zonecfg* command. By supplying a non-existent zone name to *zonecfg* we are prompted to create the new zone. The only required step when setting up a zone is to set the *zonepath* attribute. A network interface is also setup in Figure 2.5 using the *net* command.

Once the zone is created *zoneadm list* shows the zone to be in the configured state. The next step, as seen in Figure 2.6, is to install the zone.

Once the zone is installed you use *zoneadm boot* to boot the zone. There is an *autoboot* parameter that can be set in the zone configuration to have the zone autoboot on system startup. The initial boot may take a bit of time as the SMF repository is configured. After the initial boot you'll need to log into the console using *zlogin -C test_zone* and complete the final configuration.

More information on zones can be found in the *Solaris Administrators Guide: Solaris Containers, Resource Management and Solaris Zones*[3] and the man

[3] http://docs.sun.com/doc/817-1592

Figure 2.5: Zone creation

```
# zonecfg -z test_zone
test_zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:test_zone> create
zonecfg:test_zone> set zonepath=/zones/test_zone
zonecfg:test_zone> add net
zonecfg:test_zone:net> set physical=bge0
zonecfg:test_zone:net> set address=192.168.1.44
zonecfg:test_zone:net> end
zonecfg:test_zone> info
zonepath: /zones
autoboot: false
pool:
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
net:
        address: 192.168.1.44
        physical: bge0
zonecfg:test_zone> verify
zonecfg:test_zone> commit
zonecfg:test_zone> exit

# zoneadm list -cv
  ID NAME             STATUS         PATH
   0 global           running        /
   - test_zone        configured     /zones/test_zone
```

Figure 2.6: Zone installation

```
# zoneadm -z test_zone install
Preparing to install zone <test_zone>.
Creating list of files to copy from the global zone.
Copying <2379> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1157> packages on the zone.
Initialized <1157> packages on zone.
Zone <test_zone> is initialized.
The file </zones/test_zone/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

pages for *zoneadm*, *zonecfg* and *zlogin*.

DTrace

DTrace is the dynamic tracing tool built into Solaris 10. DTrace provides a tool for both developers and system administrators to analyze their running system and the applications executing on it. Because of its design, minimal impact and safety as a requirement, DTrace can be safely used on a production system. There is no need to reboot the system, restart the application or add any kind of instrumentation.

The impact of enabling DTrace is typically minimal, unless you enable large numbers of probes, and even then DTrace is designed to automatically halt any of its scripts that cause excessive performance degradation. DTrace is also designed with a dead-man in place. DTrace will kill itself if it sees performance issues arising due to its usage.

DTrace is designed around the concept of *providers* and *probes*. A provider is a high level grouping of probes. There is a default set of providers delivered with Solaris 10 including networking, function boundary tracing, application tracking and IO providers. There are also providers available for Java, Ruby, PHP and other languages. It's also possible, although not required, to write a provider for your application. Probes are the specific access points in the application, library or kernel that you can watch and gather data.

When writing a DTrace script you'll be using the *dtrace* application. As a simple starting point, Figure 2.7 is an example of using *dtrace* to get a list of all available probes and a count of the number of probes on the system. A typical machine can have upwards of 44000 probes available without looking at application probes. Since none of these probes are enabled they don't utilize any

CPU resources. As probes are enabled they are dynamically inserted into the executing code allowing DTrace to have zero performance impact when disabled.

Figure 2.7: Listing DTrace probes

```
# dtrace -l
ID   PROVIDER          MODULE                  FUNCTION NAME
 1     dtrace                                           BEGIN
 2     dtrace                                           END
 3     dtrace                                           ERROR
 4    syscall                                        nosys entry
 5    syscall                                        nosys return
 6    syscall                                        rexit entry
 7    syscall                                        rexit return
 8    syscall                                      forkall entry
 9    syscall                                      forkall return
10    syscall                                         read entry
11    syscall                                         read return
12    syscall                                        write entry
13    syscall                                        write return
...

# dtrace -l | wc -l
   44102
```

A DTrace script is made up of probes, predicates and actions. A probe is the part of the system you wish to observe, (e.g. *syscall::read:entry*). Probes are made up of four parts: *provider:module:function:name*. Predicates act like "if" conditions. Actions are what we wish to do if the probe is fired and the predicate evaluates to true.

Figure 2.8 gives an example of counting the number of syscalls the executing application is making. While this may seem a bit confusing at first glance it isn't too complicated. We use the *-n* flag to dtrace to signal that we are going to specify a probe name to trace. The probe we're interested in is *syscall:::entry*. *syscall:::entry* will match all of the probes named "entry" in the syscall provider. As you can see, if you leave part of the probe 4-tuple blank it will act as a wild-card and match everything. You can also specify the "*" as a wild-card. (e.g. open* will match open and open64). When the probe is fired we store a count into an associative array, @count, with the executable name, *execname*, as the array key.

The *count()* function is part of what is called an aggregate. You can think of aggregates like hash tables. There is a set of aggregate functions including *count()*, *quantize()* and *lquantize()*. Aggregates make it easy to pull a lot of data together without needing to store all data in memory.

Figure 2.8: Counting syscalls per application

```
# dtrace -n 'syscall:::entry {@counts[execname] = count()}'
dtrace: description 'syscall:::entry ' matched 233 probes
^C

  utmpd                         2
  snmpdx                        2
  snmpd                         2
  tsasim_exec                   3
  webservd                      3182
  appservd                      4041
  dataserver                    20187
```

More information on DTrace can be found in the *Solaris 10 Dynamic Tracing Guide*[4] and the man page for *dtrace*.

ZFS

As of Solaris 10 6/06, ZFS is available to the public. ZFS provides another option when selecting a file system to use with your Solaris machine along side other available file systems such as: UFS, VxFS and QFS.

ZFS has been designed from the ground up to be simple, secure, provide end-to-end data integrity and amazing scalability. Among its many features, because of the disk write semantics the on-disk data is always consistent, even over power failure. No need to ever run *fsck*. The on-disk data is endian agnostic, take the disk out of an x86 machine and install it in a SPARC machine and it will just work. More information on the features of ZFS can be seen in *ZFS — the last word in file systems*[5].

When working with ZFS you'll run into the terms *pools* and *file systems*. A *pool* is a collection of disks that determine the amount of storage space available. You can dynamically add and remove disks from *pools*. The disks in a *pool* can be configured in a mirrored or raid-z array.

Lets look at a couple of simple examples of creating pools and file systems. Figure 2.9 shows the simplest form of pool creation using a single disk.

Next, Figure 2.10 shows an example of creating a pool using a two disk mirror.

[4] http://docs.sun.com/doc/817-6223
[5] http://www.sun.com/2004-0914/feature/

Figure 2.9: Creating a ZFS pool

```
# zpool create tank c1t0d0
```

Figure 2.10: Creating a mirrored pool

```
# zpool create tank mirror c1t0d0 c1t1d0
```

Once the pool is created we can start creating file systems. *File systems* are created on top of storage pools. *File systems* provide a point to apply configuration parameters for different pieces of the storage *pool*. What does that mean? Well, you could have one file system with compression enabled, and one with it disabled. One could be NFS shared, one not. These attributes are also inherited as file systems are added. These properties can be things like the mount point, is the file system NFS shared, do we use compression, among others. Some properties are also inherited down the file system hierarchy. Setting compression on *tank/home* means that compression will also be enabled on *tank/home/dan*. Figure 2.11 shows the creation of a file system hierarchy and setting some properties.

Figure 2.11: Creating ZFS file systems and setting attributes

```
# zfs create tank/home

# zfs set mountpoint=/export/home tank/home
# zfs set sharenfs=on tank/home
# zfs set compression=on tank/home

# zfs create tank/home/dan
```

Because of the file system inheritance *tank/home/dan* will automatically be NFS shared and mounted to */export/home/dan*.

ZFS has built-in support for generating *snapshots*. A *snapshot* is a read-only copy of file system or volume. Creation of snapshots is almost instantaneous and since they use a copy-on-write model they'll initially consume no space in the storage pool. Figure 2.12 shows some of the snapshot commands.

Figure 2.12 is doing three things. First, we're creating a snapshot named *friday* of the *tank/home/dan* directory. Second, we can access the snapshot by changing into the *.zfs/snapshot* directory at the root of the file system. In this case, as we set the mount point to be */export/home/* we are looking at */export/home/dan/.zfs/snapshot*. We can then access the *friday* directory

Figure 2.12: ZFS snapshots

```
# zfs snapshot tank/home/dan@friday

# ls /export/home/dan/.zfs/snapshot
friday

# zfs rollback tank/home/dan@friday
```

to retrieve files saved in the snapshot. Finally, we can rollback to a previous snapshot using the *zfs rollback* command.

More information on the administration of ZFS can be found in the *Solaris ZFS Administration Guide*[6] along with the *zfs(1m)* and *zpool(1m)* man pages.

Predictive Self Healing

Tracking down the cause of, and solution too, any issues that arise on a server can be a time consuming, and error prone, task. In order to simplify the task of diagnosing and correcting these hardware and software faults Solaris 10 includes a set of technologies referred too as *Predictive Self Healing*.

Predictive Self Healing is a new capability of Solaris 10 to isolate, diagnose and recover from hardware and software faults. There are two main components to Predictive Self Healing; the Solaris Fault Manager Software and the Solaris Service Manager (SMF).

The Solaris Fault Manager Software is responsible for receiving all of the error reports in the system and taking the appropriate action. This maybe disabling a piece of errant software or an over-heating CPU.

The fault manager is made of a collection of diagnostic engines which are responsible for this diagnosis.

More information on the Predictive Self Healing capabilities of Solaris 10 can be found in *http://www.sun.com/software/solaris/availability.jsp*.

[6] http://www.opensolaris.org/os/community/zfs/docs/zfsadmin.pdf

Chapter 3

Command Differences

Most Linux commands take two types of options, short form (-v) and long form (–version). Solaris commands, unless they're the GNU version, typically don't implement the log forms. If a script is migrated from Linux to Solaris and makes use of the long option format it will, most likely, require modification to use either the GNU versions or to convert the options to their short form.

The Rosetta Stone website[1] is a good resource for finding the equivalent Solaris and Linux commands.

With the non-GNU implementation of commands there can be slight variations in the meaning and usage of options when compared to the Linux equivalent. Some of these command differences are listed below.

awk

Linux ships with *GNU awk*. Solaris ships with several versions of *awk* as listed in Table 3.1. The default version is referred to as *System V awk*. GNU awk has several extensions which aren't available with System V awk.

basename

The basic functionality of *basename* is the same between Linux and Solaris. There are two *basename* implementations on Solaris. Of these two, */usr/ucb/bin/basename* is the same as the Linux version. */usr/bin/basename* is enhanced such that you can match a suffix using a pattern as defined in *expr(1)*.

[1] http://bhami.com/rosetta.html

Table 3.1: Solaris awk variations

/usr/bin/awk	Standard System V awk
/usr/bin/nawk	New awk. Has a number of features and extensions over /usr/bin/awk
/usr/xpg4/bin/awk	XPG4 compliant awk. XPG4 awk can be used when porting awk scripts from Linux.
/opt/sfw/bin/gawk	GNU awk. The awk distribution found on the companion CD. gawk provides the highest degree of compatibility with Linux awk. The Solaris 10 Companion CD ships with GNU awk version 3.0.6.

cat

The *cat* command, while similar, has a few variations between Linux and Solaris. These differences can be seen in Table 3.2.

Table 3.2: Linux vs Solaris cat arguments

Linux	*Solaris*
-a, –show-all	-vet
–number-nonblank	-b
-e	-ve
-E, –show-ends	-ve
-s, –squeeze-blank	-
-t	-vt
-T, –show-tabs	-vt
-u (ignored)	-u (forces non buffered output.)
–show-nonprinting	-v
–help	-
–version	-

chown

Solaris ships with two versions of chown, */usr/bin/chown* and */usr/ucb/chown*. The */usr/ucb/chown* implementation only supports two options: -f and -R. */usr/bin/chown*, in addition to those supported by */usr/ucb/chown*, supports -h, -H, -L and -P. The implementations of the -f and -R flags for both commands are compatible with the Linux chown.

The -h option of */usr/bin/chown* is compatible with the -h, or –no-dereference option to chown on Linux. If a Linux script contains –no-dereference convert it

to -h when porting to Solaris.

The definition of the additional flags, -H, -L and -P can be seen in Table 3.3.

Table 3.3: Additional Solaris /usr/bin/chown arguments

-H	If the file is a symbolic link referencing a directory, the ownership of the directory and all files in the file hierarchy below it are changed. If a symbolic link is encountered, the owner of the target file is changed, but no recursion takes place.
-L	Same as -H but also affects files. Any symbolic link to a directory will be referenced and the directory traversed.
-P	If the file specified on the command line or encountered during the traversal of a file hierarchy is a symbolic link this option changes the owner of the symbolic link. This option does not follow the symbolic link to any other part of the file hierarchy. -P is similar to –no-dereference.

The implementation of chown on Linux also accepts several additional arguments. These arguments can be seen in Table 3.4.

Table 3.4: Additional Linux chown arguments

-c or –changes	Like verbose, but only reports when a change is made.
–dereference	Affects the referent of each symbolic link. This is the default behaviour in Solaris. If a script uses this option remove it when porting to Solaris.
–from=	Only change the owner and/or group of each file if its current owner and/or group match those specified. While there is no equivalent for this option in Solaris, you can use *find* with the -owner and/or -group options and pass the results to chown.

df

Solaris supports several implementations of *df*. The simplest is */usr/ucb/df* which supports an additional -v option over */usr/ucb/df*. The -v option is the same as -k except that the sizes are displayed in multiples of the smallest block size supported by each specified file system.

/usr/xpg4/bin/df supports an additional -P flag which has the same meaning as -k except the sizes are listed in 512-byte units.

du

There are several differences between the Linux *du* command and the Solaris variant. A summary of these differences is given in Table 3.5. Several *du* options available on Linux have no Solaris equivalents. These can be seen in Table 3.6.

Table 3.5: Linux and Solaris du comparison

Linux	*Solaris /usr/bin*	*Solaris xpg4*	*Solaris /usr/ucb*
-a, –all	-a	-a	a
–block-size=SIZE	-	-	-
-b, –bytes	-	-	-
-c, –total	-	-	-
-D, –dereference-args	-L	-L	-L
-h, –human-readable	-h	-h	-h
-H, –si	-	-	-
-k, –kilobytes	-k	-k	-k
-l, –count-links	-	-	-
-L, –dereference	-L	-L	-L
-m, –megabytes	-	-	-
-S, –separate-dirs	-o	-	-
-s, –summmarize	-s	-s	-s
-x, –one-file-system	-d	-x	-d
-X FILE, –exclude-from=FILE	-	-	-
–exclude=PAT	-	-	-
–max-depth=N	-	-	-
–help	-	-	-
–version	-	-	-

The meaning of the -H option differs between Linux and Solaris. On Linux, -H and -si mean the same thing, use powers of 1000 not 1024. The Solaris meaning for the -H flag is similar to that of the -L flag on Linux.

ps

The */usr/ucb/ps* executable is BSD compatible and takes the same command line arguments as the Linux version of *ps*. The output on Solaris may differ from the output on Linux.

Table 3.6: Linux du options without Solaris equivalents

-b	print size in bytes
–block-size=SIZE	
-c	produce a grand total
-l	count links
-L	dereference all symbolic links
-m	megabytes
–exclude=PAT	
–max-depth=N	
–help	
–version	

setfacl

The */usr/bin/setfacl* command is used for managing file Access Control Lists (ACL). Although they perform the same function in Linux and Solaris, the syntax and options are different.

The syntax for the Solaris implementation can be seen in Figure 3.1.

Figure 3.1: Solaris setfacl Syntax

```
setfacl [-r] -s acl_entries file
setfacl [-r] -md acl_entries file
setfacl [-r] -f acl_file file
```

The -s option sets a file's ACL. All old ACL entries are removed and replaced with the newly specified ACL.

The -m option adds one or more new ACL entries to the file, and/or modifies one or more existing ACL entries on the file. It can also be used to replace an existing ACL entry.

The -d option deletes one or more ACL entries from the file.

The -r option recalculates the ACL mask entry.

The format for an ACL entry is similar to the format in Linux. Linux supports the Solaris ACL entry format, while some Linux formats are not supported by Solaris. The ACL formats seen in Figure 3.2 are supported by Solaris.

Linux accepts an additional colon (:) after the "other" and "mask" keywords, while Solaris does not.

The *perms* are composed of the symbols "rwx" or a number (the same permission numbers used with the *chmod* command). For example "r-x" is used

Figure 3.2: Solaris ACL formats

```
[d[efault]:][u[ser]]:uid:perms
[d[efault]:][g[roup]]:gid:perms
[d[efault]:]o[ther]:perms
[d[efault]:]m[ask]:perms
```

to signify read and execute permissions. Linux support "rx" as an equivalent, while Solaris treats this as an error.

Solaris *setfacl* supports the -f option which is absent in the Linux version of *setfacl*. This option allows you to use another file's ACL as a reference for setting a file's ACL.

The *setfacl* options listed in Table 3.7 are supported in Linux but have no equivalent in Solaris.

Table 3.7: Linux setfacl options not available in Solaris

-b, –remove-all	removes all ACL entries
-k, –remove-default	removes the default ACL
-n, –no-mask	Do not recalculate the effective rights mask. This is equivalent to the Solaris setfacl without the -r option. The absence of this option is equivalent to specifying the -r option in Solaris.
–mask	Recalculate the effective rights mask. Equivalent to the -r option in Solaris
-d, –default	All operations apply to the Default ACL
–restore=file	Restore permission backup created by "getfacl -R" or similar
–test	test mode
-R, –recursive	Apply operations to all files and directories recursively
-L, –logical	Logical walk, follow symbolic links
-P, –physical	Physical walk, skip all symbolic links
–version	Print the version of setfacl and exit
–help	Print help explaining command line options
–	End of command line options
-	If the file name parameter is a single dash, read a list of files from standard input

getfacl

The */usr/bin/getfacl* command is used to display the Access Control Lists(ACL) of files. Although they perform the same function and have similar output formats in Linux and Solaris, they differ in the options they accept.

None of the long form options are supported under Solaris. In addition to that, the following short form options are also not supported: -R, -L, -P, and -. These options do not have any equivalents in Solaris.

tar

The implementation of *tar* included in Linux is GNU tar. Solaris uses System V tar. These two implementations have many differences. It is best to consult their respective man pages rather than providing a lengthy discussion of these differences in this guide.

Notably missing in Solaris are the compression options (-Z, -z, -j). Solaris *tar* does not support the use of an external compression programs.

Solaris does include an implementation of GNU tar in */usr/sfw/bin/gtar*. GNU tar is installed via the SUNWgtar package.

When porting Linux scripts that use *tar* to Solaris, you can either re-write the *tar* command to use the equivalent Solaris *tar* options and use pipes to the compression and decompression programs. Alternatively, if the target Solaris system has the SUNWgtar package installed, the script can be modified to use *gtar* instead of *tar*.

useradd

The *useradd* command in Linux is very similar to the same command on Solaris. For the most part they operate identically and take most of the same options which mean the same things. The Solaris version takes additional options to support RBAC (see Hardening Tools for more information). One significant difference is the meaning of the -p option. In Linux, this is used to specify a password for the account on the command line; an unsafe practise that the Linux version supports. In Solaris the -p option is used to specify the project name to which all processes started by the user being created will belong.

Table 3.8 details the option differences between Solaris and Linux.

For more information on the Solaris *useradd*, consult the useradd(1M) man-page.

Table 3.8: Linux and Solaris useradd differences

Linux	*Solaris*	*Meaning*
-e expire_date	-e expire_date	User account expiration date. Format of expire_date differs between Linux and Solaris.
-f inactive	-f inactive	On Linux this is the number of days after password expiry until the the account is permanently disabled. In Solaris this is the maximum number of days allowed between uses of a login ID before considering it invalid.
-M		The users home directory will not be created.
	-m	Users home directory will be created if it doesn't already exist.
-n		Create a group having the same name as the user. In Solaris, use a separate call to groupadd
-o	-o	Allow the user to be created with non-unique user id.
-p passwd		Specify the encrypted password as returned by crypt(3).
	-p profile	Specify a project to which the user belongs.
-r		Create a system account (a user with a UID lower than the value of UID_MIN defined in /etc/login.defs). This is a RedHat specific option.
	-K key=value	Set a key=value pair to add to the users attributes.
	-A authorization	Authorizations to add to the user.
	-P profile[,profiles...]	Profiles to add to the user.
-D -e default_expire_date	-D -e default_expire_date	Set the default expiry date for accounts created. Format of default_expire_date differs between Linux and Solaris.

groupadd

The *groupadd* command in Linux is very similar to the same command on Solaris. For the most part they operate identically and take most of the same options with the same meanings. Table 3.9 describes the differences between the implementations.

Table 3.9: Linux and Solaris groupadd differences

Linux	*Solaris*	*Meaning*
-r		Instructs groupadd to add a system account.
-f		Force option. This is a RedHat specific option.

For more information on the Solaris *groupadd*, consult the groupadd(1M) manpage.

Chapter 4

Installation

This chapter will talk about three installation mechanisms, CD/DVD installation, network installation and flash archives. We will also look at a mechanism to upgrade an existing machine, live upgrade.

After reading this chapter you should be able to install Solaris onto an existing machine.

Topics Covered
CD/DVD Media Installation
Network Installation
Flash Archives Live Upgrade

CD/DVD Media Installation

The most typical installation method for both Linux and Solaris is using the provided, or downloaded[1], installation media.

Installation of Linux typically involves the following steps:

- Boot from CD.
- Partition hard drives.
- Select desired software packages.
- Enter system configuration parameters.

A Solaris install will follow a similar pattern:

- Boot from CD/DVD.
- Enter system configuration parameters.
- Select desired software packages.
- Partition the hard drives.

There is nothing overly complex involved in a media based installation. Once the required information is entered the system will be rebooted.

Network Installation

Unlike a media installation, the steps for a network installation differ between Linux and Solaris. On Linux you would boot off a CD/DVD and then choose the install media for your packages to be the URL of the host server. The install program downloads the packages and installs them to the local system.

Solaris uses a technology called Jumpstart. You'll need a Jumpstart server with a DHCP service running. The Jumpstart server will contain a copy of the installation media (CD or DVD) on an NFS exported file system.

You'll need to know the MAC address of the host to be installed in order to configure the Jumpstart server. Our example will use 00:0c:29:d5:c1:90.

First, we configure the Jumpstart server. This involves mounting the Solaris install media and executing the *setup_install_server* {*install directory*} script. The *install directory* parameter is the directory to copy the install image to on

[1] Solaris 10 is available at: http://www.sun.com/software/solaris/get.jsp

Figure 4.1: DHCP configuration

- Default settings for everything except as below.
- Text file as the data store.
- /var/dhcp as location for data store.
- Do not manage host records.
- Do not provide DNS domain nor name server information.
- Network used is 172.16.129.0, netmask 255.255.255.0.
- Network type is LAN and routing is using Router discovery protocol.
- Do not provide NIS information.
- Do not provide NIS+ information.
- Starting address for DHCP clients is 172.16.129.100 – with 10 addresses

the Jumpstart server. *setup_install_server* is available in the *Solaris_10/Tools* directory of the install media. If you're using CDs you'll be prompted to enter the additional CDs as required.

With the Jumpstart server installed we need to configure the DHCP server. This is done by executing */usr/sadm/admin/bin/dhcpmgr* as root. This will prompt you to enter the configuration for your DHCP server. An example DHCP configuration is provided in Figure 4

If you execute *svcs -a |grep dhcp* you should see that *dhcp-server* is enabled.

The last action is to tell the Jumpstart server and DHCP server about the client to be installed. This is done by issuing *add_install_client -d -e 00:0c:29:d5:c1:90 i86pc* (assuming you're installing an x86 machine). If NFS or TFTP are not enabled then *add_install_client* will configure them as needed. You can verify these services are executing by using *svcs -a |grep nfs* and *inetadm |grep tftp*.

Follow the instructions provided by *add_install_client* to configure the DHCP server, add a macro 01000C29D5C190 and assign:

- Boot server IP (BootSrvA) : 172.16.129.10
- Boot file (BootFile) : 01000C29D5C190

Then, using *dhcpmgr*, assign this macro (01000C29D5C190) to the DHCP address range. With that, everything is configured.

Depending if you're booting a SPARC or x86 machine the boot actions are slightly different. On SPARC systems, the install client is booted from the network and retrieves its hostname using *rarp*. It then retrieves and executes a boot file. The boot file obtains the boot parameters from bootparamd. On x86 systems, the DHCP server will tell the x86 PXE client the bootservers IP and the boot file information. The root file system for the install client will be mounted from the NFS server. The kernel is then loaded from this mounted file system. With the kernel loaded the installation parameters will be obtained.

More information on network installations of Solaris 10 can be found on page 119 of in the *Solaris 10 Installation Guide: Network-Based Installations*[2]. Information on customizing and automating Jumpstart can be found in *Solaris 10 Installation Guide: Custom JumpStart and Advanced Installations*[3]. Information on booting and installing Solaris over the Internet (WANboot) can be found in *Solaris 10 Installation Guide: Network-Based Installations Part III* [4].

Flash Archives

If you have several systems that are clones of each other, similar configuration and application installs, you can use Flash Archives to make installation simpler.

A flash archive is a file that contains all the files from a reference system. The reference system will have been previously installed and configured as required. All machines installed from the flash will be identical to the reference system, a virtual clone of the master system.

More information on flash archives can be found in *Solaris 10 Installation Guide: Solaris Flash Archives (Creation and Installation)*[5].

Live Upgrade

Live upgrade provides the ability to create alternate boot environments on a Solaris host. You can then install updates into the alternate environment while the current boot environment continues to execute. When the install is complete, the system can be rebooted to the alternate boot environment.

Using live upgrade can reduce the downtime for installation and upgrade to minutes. It also provides the ability to fall back to the previous boot environment in case of problems resulting from the upgrade.

More information on live upgrade can be found in *Solaris 10 Installation Guide: Solaris Live Upgrade and Upgrade Planning*[6].

[2] http://docs.sun.com/app.docs/doc/819-5776
[3] http://docs.sun.com/app/docs/doc/819-5778
[4] http://docs.sun.com/app/docs/doc/819-5776/6n7r9js52?a=view
[5] http://docs.sun.com/app/docs/doc/819-5779
[6] http://docs.sun.com/app/docs/doc/819-5777

Chapter 5

Software Management

Any operating environment will require applications to be installed, and patches applied. This chapter takes a look at the mechanisms to install packages on Linux and Solaris and touches on the issue of patching a machine.

Topics Covered
RPM Packages
Solaris Packaging
Patching

RPM packages

Package management in RedHat or SUSE Linux is handled by the *rpm* command. Package installation is facilitated using *rpm -i* and package removal with *rpm -e*.

Solaris Packaging

Solaris uses System V packages. To install packages you'll use *pkgadd*. Removal is handled by *pkgrm*. Solaris also provides support for the *rpm* command.

Patching

In Linux, the concept of patching doesn't exist as it does in Solaris. In Linux you would upgrade the given RPM to the next revision. There is no built in mechanism in Linux to rollback a patch after it has been applied.

In Solaris you can use the *patchadd* and *patchrm* commands to handle the application and removal of patches. Patches are available on the SunSolve[1] website. While some patches require a support contract, critical patches are available for free. Each patch file contains all of the changes that must be applied, or rolled back, as a bundle.

On a RedHat system you can use *up2date* to handle any package version changes required to update to the latest revision of a package.

There are two separate commands available on Solaris to handle these upgrades. *updatemanager* provides a GUI environment to manage the application and rollback of patches, while the *smpatch* application provides a command line interface to the same functionality.

[1] http://sunsolve.sun.com

Chapter 6

System Management

The management of a Solaris system is quite similar to that of a Linux system. Often the commands used are the same, or, in the case of printing, the subsystem is the same. This can aid in the move from Linux to Solaris as the experience gained on Linux is directly applicable on Solaris.

This chapter will attempt to go through some of the aspects of system management and highlight some of the similarities and differences between Linux and Solaris.

Topics Covered
Booting, Shutdown and Run Levels
System Services
User/Group Management
Printer and Printing Management
File System Management
Disk and Volume Management
Network Management
Remote Management
Kernel Configuration

Booting, Shutdown and Run Levels

Up to Solaris 9 the boot procedures between Linux and Solaris were very similar. Both systems provided the concept of *run levels* that were used to define what services were started and stopped at each level. The *init* command was used to switch between these run levels.

Table 6.1 lists the traditional run levels seen on a Linux system.

Table 6.1: Linux run levels

0	halt
1	single user
2-5	multi user
6	reboot

Solaris machines up to Solaris 9 used a set of run levels as shown in Table 6.2.

Table 6.2: Solaris run levels before Solaris 10

S, s or 1	single user state
0	halt
2	multi-user no network services
3	multi-user with network services
4	unused
5	power off
6	reboot

With Solaris 10, and the inclusion of SMF, the run levels have been replaced by milestones. Table 6.3 lists the default set of milestones.

Table 6.3: Solaris 10 milestones

sysconfig	
devices	
single-user	Equivalent to run level 1.
network	
name-services	
multi-user	Equivalent to run level 2 in previous Solaris versions.
multi-user-server	Equivalent to run level 3 in previous Solaris versions.

Solaris 10 still provides a limited set of run levels as seen in Table 6.4. As in

previous versions of Solaris, and with Linux, you can use *init* to switch to any one of these levels.

Table 6.4: Solaris 10 run levels

0	halt
1, s, or S	single user
5	power off
6	reboot

Two other commands are available to manipulate the current system state on Solaris, the *reboot* and *halt* commands. These commands will restart or halt the system, respectively. It should be noted that when using these commands the system will not go through the normal shutdown procedures. Services are not stopped. Processes are simply killed, file systems unmounted and the system rebooted or halted.

The recommended method to reboot or halt the system is to use *init*.

System Services

When using a Linux system, unless run from *inittab* with the respawn attribute, system services will not be respawned if they're killed or terminate abnormally. With Solaris 10, and the addition of SMF, it isn't possible to simply kill a system service as it will be automatically restarted by SMF. You have to use the *svcadm* commands to disable or enable the service.

Services that would be handled by *xinetd* on Linux are managed by SMF in Solaris 10. When editing the services available through inetd you will edit the */etc/inet/inetd.conf* file. You must then execute *inetconv* to create the corresponding service entries in SMF. All control of the inetd services will now be done through *inetadm* or *svcadm*.

On a Linux system the typical places to configure services are listed in Table 6.5.

Table 6.5: Linux services configuration locations

/etc/inittab	To be controlled by init.
/etc/rc*.d	Run level specific scripts that start system services.
/etc/(x)inetd.conf	Controlled by inetd.

On Solaris those locations are listed in Table 6.6.

Table 6.6: Solaris services configuration locations

/etc/inittab	To be controlled by init. Not recommended in Solaris 10.
/etc/rc?.d, /etc/init.d	Run level specific scripts that start system services.
/etc/inetd.conf	Controlled by inetd; in Solaris 10, use SMF and inetadm.
SMF	Solaris 10 only.

User/Group Management

Solaris extends the *useradd* and *groupadd* commands as seen in Linux to provide extensions to manage RBAC-related properties. Other options may differ between Linux and Solaris.

Along with *useradd* and *groupadd* Solaris also provides *smuser* and *smgroup* to manage accounts and groups in a name server, such as NIS. These two commands are part of the Solaris Management Console (SMC). *smc* is also available as a graphical environment to manage users and groups.

Printing and Printer Management

Most Linux systems provide CUPS to handle their printing and printer management. Solaris 10 includes CUPS as well, making it compatible with Linux systems.

Prior to Solaris 10, Solaris used System V printing. The System V printing system provided the commands given in Table 6.7 to interact with the print system.

Table 6.7: System V print commands

lpadmin	Modify printing system parameters.
lpsched	Start the print server (/usr/lib/lp/lpsched).
lpshut	Stop the print server.
cancel	Cancel print jobs.
lpmove	Move print jobs to another printer.
lp	Submit a print job.
lpstat	Display the status of printers and/or print jobs.

Print system configuration was stored in */etc/printers.conf*, printers NIS map, the users *$HOME/.printers*, the $PRINTER and the $LPDEST environment variables.

A graphical interface for setting up printers in Solaris 10 is *printmgr*. This is similar to the GNOME printer management command *gnome-cups-manager* available on most Linux systems. The *printmgr* command resides in */usr/sadm/admin/bin/printmgr*. */usr/sbin/printmgr* exists as a symbolic link to it.

File System Management

Solaris supports a wide variety of file system types to support most storage media (CDs, DVDs, Hard Drives, floppy disks, flash based storage) and network based file system protocols. Solaris also uses file systems to implement various system interface features, and to export some kernel information as files visible to the user (ie. /etc/mnttab). Table 6.8 shows the various file system types supported under Solaris 10. In addition to the native supported file system, third party software vendors also provide file systems; for example, Veritas provides the *vxfs* file system.

Table 6.8: File System Types

autofs	Automount File System.
cachefs	Caching File System.
ctfs	Contract File System.
devfs	Devices File System.
fd	File Descriptor File System.
hsfs	High Sierra File System (CDs).
lofs	Loopback Virtual File System.
mntfs	Mount Table File System (/etc/mnnttab).
nfs	Network File System.
objfs	Kernel Object File System.
pcfs	DOS formatted file system.
proc	/proc File System.
qfs	Distributed file system.
sam-fs	Archive management and retrieval.
tmpfs	Memory based file system (/tmp).
udfs	Universal Disk Format File System (DVDs).
ufs	Unix File System.
volfs	Volume Management File System.
xmemfs	Extended Memory File System.

In Solaris, file systems are mounted using the *mount(1M)* command. Linux places the mount command in */bin*. In Solaris this command is located in */usr/sbin*. The file system type is provided to the Linux *mount* command using the *-t vfstype* option. The equivalent option for the Solaris mount command is *-F FSType*.

Loopback Devices

Loopback devices provide a mechanism that allow mounting of disk images as file systems. (They are also used in Solaris to handle the loopback file system mounts in zones.)

When mounting a loopback device on Linux you would use a command similar to *mount -o loop /path/to/disk/image /mountpoint*. This would mount the image */path/to/disk/image* to the directory */mountpoint*.

On Solaris, instead of mounting the disk image directly you'll use *lofiadm* to create a loopback device that will then be mounted. Execute *lofiadm -a /path/to/disk/image* which will produce a */dev/lofi/X* device. This device is then used to mount the file system, *mount -F FSType /dev/lofi/X /mountpoint*. You will need to provide the proper -F FSType option to the Solaris mount command. The FSType provided should match the file system found in the disk image. For example if the disk image is an ISO image of a CD, you would use *mount -F hsfs /dev/lofi/X /mountpoint*.

File System Quotas

There are three main commands to work with file system quotas on Solaris. Of the three, only *quot* is unique to Solaris. *quot* will list the file system usage per system user.

The other two commands, *edquota* and *quota*, while being named the same as their Linux counterparts, have differing options and behaviour.

Disk and Volume Management

Disk Management

There are two main commands used to work with disks on Solaris, *fdisk* and *format*. The *fdisk* command (on x86 systems) is used to create a partition on the disk. Solaris partitions have type 0x82, which is the same as the Linux SWAP partition. This may cause issues when dual booting an x86 machine with Solaris.

Solaris uses a single partition with type 0x82 then uses Sun disk labels within the partition to split it further into slices. This slicing is done with the *format* command.

As of Solaris 10 6/06 the partition type 0x82 is depreciated. A new type, 0xbf, is now used as the partition type for Solaris (this is the Solaris2 type). Solaris will continue to recognize the older 0x82 (solaris) ID. Older non-Solaris partitioning software may not yet recognize the 0xbf partition type.

format is used to slice up Solaris fdisk partitions (on x86 systems) or disks (SPARC) into Solaris slices. *format* will present the user with a list of disks to manage. All disks recognized by the system are listed. Once a disk is selected, the *part* command can be used to partition it into slices.

Volume Management

Volume management on Linux is controlled using the *vg** and *lv** commands.

If you aren't using ZFS, then Solaris uses a set of metadevices for volume management. These metadevices are maintained using the meta* commands given in Table 6.9.

Table 6.9: Metadevice commands

metadb	Creates and manages meta device databases.
metainit	Initializes metadevices.
metattach	Used to attach a metadevice to a mirror and to attach space to a soft partition.
metadetach	Used to detach a metadevice from a mirror.
metahs	Used to manage hotspare pools.
metaoffline	Place submirrors offline.
metaonline	Place submirrors online.
metaparam	Modify parameters of metadevices.
metarecover	Recover soft partition information.
metarename	Renames a metadevice.
metareplace	Enable or replace components of submirrors or RAID5 metadevices.
metaset	Configure shared disksets.
metastat	Display status of metadevice or hot spare pool.
metaclear	Delete active metadevices and hot spare pools.
metadevadm	Updates metadevice information.
metasync	Performs metadevice resync during reboot.

Network Management

There are commands, as seen in Table 6.10, and files, as seen in Table 6.11, which are involved in the configuration of networking on a Solaris host.

Table 6.10: Networking tools

ifconfig	Configure interfaces and devices.
route	Configure network routes.
netstat	Display network configuration and connection status.
ndd	Get/set configuration parameters in select kernel drivers.

Table 6.11: Networking configuration files

/etc/hostname.[interface_name]	Each network interface has a corresponding file which indicates the hostname to use for assigning IP addresses to the interface. Also used to indicate that a network interface is to use DHCP.
/etc/nodename	Hostname of the system.
/etc/hosts	Static table of IP addresses.
/etc/defaultrouter	IP address of the default router.
/etc/netmasks	Table of default netmasks for various networks.
/etc/networks	Table of networks and their names.
/etc/dhcp.[interface_name]	DHCP parameters for interface.
/etc/resolv.conf	DNS client configuration.
/etc/nsswitch.conf	Name service switch file. Used to select the source for various network information parameters.

Remote Management

Solaris comes bundled with both *telnet* and *ssh* to handle remote system access. These services can be enabled and disabled with SMF as needed.

In addition to the command line access you can also use the SMC GUI to manage a remote Solaris host.

Kernel Configuration

Making kernel configuration changes differs between Solaris and Linux. On a Linux system you may need to modify the source, do runtime modifications to entries in /proc, use sysctl or load kernel modules. On Solaris you may need to modify /etc/system, load kernel modules, run utilities such as ndd, use DTrace or adb.

/etc/system is the kernel configuration file. This file is loaded at boot time by the kernel and the kernel behaviour is influenced by the settings in this file. You can influence general kernel behaviour, including paging, swapping, process sizing, file system flushing, kernel memory allocation, scheduling, TCP/IP parameters, among others. You can also force certain kernel modules or device drivers to be loaded at boot time. Parameters passed to device driver modules can also be configured by setting up values in /etc/system. Since this file is only read once at boot time, changes do not take effect until after a system reboots.

For more information on kernel tuning using /etc/system consult the *system(4)* man page and the *Solaris Tunable Parameters Reference Manual*[1].

Loading Kernel Modules

Kernel modules are binary files that contain kernel code. These usually implement some sort of device driver, file system, system call, or some other kernel level functionality. These modules may be loaded and unloaded throughout the lifetime of the OS. The kernels functionality can be modified by loading in a module. For example a new file system can be supported by loading in a kernel module that implements that file system. The command to load kernel modules is *modload*. To remove a kernel module from the running kernel, the *modunload* command is used. The *modinfo* command allows the user to view what modules are currently loaded on the system. These commands are the Solaris counterparts to the *modprobe*, *insmod*, *rmmod*, and *lsmod* commands on Linux. See *modload(1M)*, *modunload(1M)*, and *modinfo(1M)*.

[1] http://docs.sun.com/app/docs/doc/819-2724

Kernel tuning Commands

There are some tuning parameters that can be modified using commands. For example, *ndd* is a utility for modifying the behaviour of network interfaces. With *ndd* you can change the configuration of your network interface (from half duplex to full duplex, for example). You can also change the way the TCP/IP stack in solaris behaves. Consult *ndd(1M)* for details.

In the previous section /etc/system was mentioned as a way to modify kernel variables. However /etc/system modifications required a system restart to take effect. *adb* and *dtrace* are utilities that can allow you to directly modify kernel parameters while the system is running. Their effects are immediate. These should be used with care as errors could prove to be fatal to the running kernel.

More information on kernel tuning for Solaris can be found in the *Solaris Tunable Parameters Reference Manual*[2].

[2] http://docs.sun.com/app/docs/doc/817-0404

Chapter 7

Device Management

Adding and removing devices from machines has become commonplace with the advent of USB drives, external hard drives, digital cameras and other portable devices. This chapter takes a look at how Solaris handles these devices.

Topics Covered
Device Naming and Access
Adding/Removing Devices
Removable Media
Tape Drives
Terminals, Modems and Serial Ports

Device Naming and Access

On Linux the */dev* directory stores all of the device files needed for the system. There is also a selection of TTY devices named */dev/pty**. Each disk available on the system will appear in */dev* with a name of */dev/sd[a-z]* for SCSI disks and */dev/hd[a-z]* for IDE disks. Partitions are indicated by */dev/sdaN* where N is the partition number. The */dev* directory can be considered flat, with all of the device nodes at the save level in the directory.

Although similar, the */dev* directory on Solaris has some significant variations. */dev* on Solaris does not contain any device files. Instead, all entries are symlinks to the */devices* directory. Solaris */dev* is also set out in a hierarchy. There are sub-directories for different types of devices: dsk, rdsk, pts, cua and rmt.

Disk and TTY devices are also named slightly differently. TTYs are named using the form */dev/pts/**. Solaris uses controller, target, device, slice to address partitions on a disk. (e.g.*/dev/dsk/cAtBdCsD* where A is the controller number, B is the SCSI target ID, C is the lun, and D is the slice number with 0 as the first slice.)

There are no devices on Solaris that point to the disks themselves. All disk devices point to a slice of the disk. Slice 2, typically, is a special slice that overlaps all other partitions starting from cylinder 0 and encompassing the entire disk.

Adding/Removing Devices

Linux uses *modload* and *modunload* to add and remove devices. A static kernel will require that the device driver has been compiled into the monolithic kernel and initialized at boot.

On Solaris 8 and older systems you would use the *adddrv* command to add and remove devices. With Solaris 9 and newer systems the *devfsadm* command is used. *devfsadm -C* can be used to clean up stale */dev* entries. The */devices* tree reflects what the OpenBoot Prompt (OBP) saw at system boot time.

As with any Linux distrobution the latest Solaris release will have the most comprehensive driver support available.

Removable Media

Removable devices in Solaris are controlled by the volume manager *vold*. *vold* is started at boot using */etc/init.d/volmgt*.

When floppies are inserted they are automatically mounted to */floppy* and create two devices: a block device, */vol/dev/diskette0*, and a raw device, */vol/dev/rdiskette0*.

CDs and DVDs work in a similar fashion to floppies. They are automounted to */cdrom* and create device nodes in */vol/dev/dsk* and */vol/dev/rdsk* for block and raw access respectively.

Tape Drives

SCSI tapes are represented by device files in */dev/rmt*. The device files have the form of */dev/rmt/N[lmhuc][b][n]* so follows:

N	The device number, 0 for first tape on the system.
lmhuc	Density (low, medium, high, ultra/compressed).
b	BSD behavior (whether the fsb or fsf mt commands point to the end of the previous file (fsb), or the beginning of the current file (fsb), end of current file (fsf), or beginning of next file (fsf)).

Terminals, Modems and Serial Ports

There are a couple of typical Linux tools for working with terminals and modems connected to serial ports. They are *minicom* and *seyon* for working with ports and *setserial* for working with serial ports.

The main Solaris tool is *tip* which is used to connect to serial ports. *tip* can be configured via */etc/remote* and the *$HOME/.tiprc* file.

Serial port speed, parity and handshaking is performed via the *eeprom* command.

Chapter 8

Security and Hardening

Any system that is hooked up to a network needs to be secured. Security is a multi-faceted task, from the initial hardening to auditing for patches and vulnerabilities.

This chapter presents some of the options for hardening and auditing a Solaris machine.

Topics Covered
Hardening Tools
Least Privileges
Auditing Tools
Securing and Removing Services
Kernel Tuning for Security

Hardening Tools

There are several different tools available for the hardening of a Solaris machine. These facilities range from TCP Wrappers, included in Solaris 9 and 10 with the SUNWtcpd package, to allow for the tight control of incoming TCP connections to the RBAC and BART tools listed below.

Solaris Security Toolkit

The Solaris Security Toolkit (previously know as the Jumpstart Architecture and Security Scripts, JASS), provides an automated, extensible and scalable mechanism to maintain a secure Solaris system. The JASS scripts allow you to audit and harden your Solaris installation.

More information on JASS can be found at: *http://www.sun.com/software/security/jass/*.

RBAC

The Role Based Access Control (RBAC) facility allows the system administrator to assign roles and privileges to users on the system. This allows the admin to provide access to facilities of the system that would traditionally require root access.

More information on RBAC can be found in *rbac(1)* and in *http://www.sun.com/bigadmin/features/articles/least_privilege.html*.

BART

The Basic Auditing and Reporting Tool (BART) is designed to monitor your file system and look for changes in file contents. BART does this by looking at all of the files on the system and recording information about each one.

BART, although less robust, is similar to the commercially available Tripwire application.

More information on BART can be found in *bart(1M)*.

Least Privileges

There are typically a large number of processes executing on a UNIX box with full root privileges as they need to access devices, modify other processes, work with restricted files or access other root restricted resources.

Having all of these processes with root privileges around can pose a security risk. If any of those applications is hacked the user could gain full root privileges on the system.

This is where the Least Privileges capabilities of Solaris 10 come into play. The least privileges feature allows you to remove any system privileges from an application that are not required for its normal operation. There are around 50 privileges that can be set on a process.

More information on the least privileges capabilities of Solaris 10 and the available privileges can be found in *privileges(5)* and *http://www.sun.com/bigadmin/features/articles/least_privilege.html.*

Auditing Tools

Many of the auditing tools available on Linux are also available on Solaris. This includes *tripwire*[1], used to monitor changes to important files. *crack*[2] and *John the Ripper*[3] are also available to check for weak passwords..

Securing and Removing Services

The processes of hardening a Linux or Solaris 9 and prior machines is actually quite similar. For any service you don't wish to start you just remove the relevant files from */etc/rc*.d.* For any inet files you'll need to remove them from the inetd configuration. On Solaris this file would be stored at */etc/inetd.conf.*

With Solaris 10, for any service to be disabled you will use the *svcadm disable* {*service name*} and *inetadm -d* {*service name*} to set them as disabled in the SMF repository. See Section 2 for more information on SMF.

Kernel Tuning for Security

Solaris provides two ways to guard against stack based buffer overflows. The first is done globally by setting *noexec_user_stack* into */etc/system.* All applications that run will have the option set. If you don't wish to do this globally an individual application can link with the */usr/lib/ld/map.noexstk* map file.

More information on the *noexec_user_stack /etc/system* option can be found in: *Solaris Tunable Parameters Reference Manual* [4].

[1] http://www.sun.com/software/security/tripwire/
[2] ftp://ftp.cert.dfn.de/pub/tools/password/Crack/
[3] http://www.openwall.com/john/
[4] http://docs.sun.com/app/docs/doc/817-0404/6mg74vs9h?a=view

Chapter 9

Monitoring and Performance

It is often necessary to monitor your systems to determine the CPU, memory, disk or network load. There are several tools available on Solaris to aid in this monitoring and analysis. Along with the tools listed in this chapter there are several resources for DTrace scripts available that can aid system monitoring and performance tuning. The OpenSolaris DTrace community[1] is a good source for these scripts.

This chapter will look at a few of the available tools and their usage.

Topics Covered
Processors
Memory
Network
Disks, Volumes and File Systems
System and User Processes
Input/Output

[1] http://opensolaris.org/os/community/dtrace/scripts/

Processors

Information on the current processors in the system can be retrieved using the *psrinfo* command. An example of the *psrinfo* output can be seen in Figure 9.1.

Figure 9.1: psrinfo and psrinfo -v

```
# psrinfo
0       on-line   since 09/28/2006 19:49:33

# psrinfo -v
Status of virtual processor 0 as of: 09/29/2006 02:06:34
  on-line since 09/28/2006 19:49:33.
  The i386 processor operates at 1733 MHz,
        and has an i387 compatible floating point processor.
```

The *mpstat* command can be used to gather per-processor statistics on the system. The output data will contain one line per processor. Figure 9.2 shows example output from *mpstat*.

Figure 9.2: mpstat 1 3 output

```
# mpstat 1 3
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0 1305   3    0   394  294  403   38    0    0    0  1554    7   3   0  90
  0   10   0    0   358  258  205    0    0    0    0   250    1   1   0  98
  0    0   0    0   345  245  186    0    0    0    0   195    1   0   0  99
```

A field of note in the *mpstat* output is *xcal*, if you have multiple CPUs. This will be the number of inter-processor cross-calls. *csw* is the number of context switches. *syscl* is the number of system calls that happened on the processor. *usr*, *sys* and *idl* give the amount of user time, system time and idle time respectively. The *wt* column is a legacy artifact and will always return 0 on Solaris 10.

The *kstat* command can also be used to gather information on the processors. By providing the *-m cpu* option to *kstat* you will retrieve all the CPU information. Figure 9.3 shows an example of this usage.

Memory

The *vmstat* command can be used to display information on the system's virtual memory subsystem. *vmstat* gives information on the current swap and memory,

Figure 9.3: kstat -m cpu output

```
# kstat -m cpu
module: cpu                             instance: 0
name:   intrstat                        class:    misc
        crtime                          6.573712134
        level-1-count                   1214737
        level-1-time                    3558707908
        level-10-count                  2364949
...
```

page faults, disk stats and fault information. Figure 9.4 shows example *vmstat* output.

Figure 9.4: vmstat 2 5

```
# vmstat 2 5
 kthr      memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr cd f0 s0 --   in   sy   cs us sy id
 0 0 0 1998572 1101884 8 29 14  0  1  0 35  8  0  0  0  397  865  412  5  1 94
 0 0 0 1913292 1014728 0 22  6  0  0  0  0  0  0  0  0  364  204  207  1  0 99
 0 0 0 1913292 1014728 0  0  0  0  0  0  0  0  0  0  0  356  162  184  1  0 99
```

kstat can be used to gather memory information. The memory module is *vmem*. Figure 9.5 shows example output.

Figure 9.5: kstat -m vmem

```
# kstat -m vmem
module: vmem                            instance: 1
name:   heap                            class:    vmem
        alloc                           4632
        contains                        0
        contains_search                 0
        crtime                          0
```

Network

The *netstat* command can be used to gather network status information. *netstat* can retrieve information on the active network sockets, various network data

structures, STREAMS memory statistics, interface states, routing tables and DHCP information.

You can use *kstat* to query information on each network interface driver in the system. In my case, I'm using the *bge* driver. Figure 9.6 is an example of the output.

Figure 9.6: kstat -m bge

```
# kstat -m bge
module: bge                             instance: 0
name:   bge0                            class:    net
        brdcstrcv                       16059
        brdcstxmt                       14
        collisions                      65
        crtime                          128.886063641
        ierrors                         0
        ifspeed                         100000000
        ipackets                        33976
```

Disks, Volumes and File Systems

The *df* command displays information on the size, used and available space of the mounted file systems. Figure 9.7 shows example *df* output.

Figure 9.7: df -h

```
# df -h
Filesystem         size   used  avail capacity  Mounted on
/dev/dsk/c0d0s0    9.6G   3.1G   6.4G    33%    /
/devices             0K     0K     0K     0%    /devices
ctfs                 0K     0K     0K     0%    /system/contract
```

The *du* command can be to summarize disk usage information. *du* will output the size of each file and each directory as it walks the file system from the given starting directory.

metastat can be used to display metadevice or hot spare pool information.

System and User Processes

System information can be gathered with the *prstat* command. *prstat* shows process and thread information. *prstat* is similar to the *top* command. *top* is also available for Solaris as part of the Companion CD. Figure 9.8 shows example output from *prstat*.

Figure 9.8: prstat

```
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
533 dj2        36M   62M sleep   59    0   0:03:45 3.8% Xorg/1
627 dj2        62M   17M run     49    0   0:00:16 0.5% gnome-terminal/2
577 dj2        17M   11M sleep   59    0   0:00:41 0.3% enlightenment/1
873 dj2      3972K 2740K sleep   59    0   0:00:00 0.2% vim/1
```

Input/Output

The *iostat* command can be used to display information on the different IO devices in the system. This can include hard drives, TTY devices, NFS mounts and other devices. Example *iostat* output can be seen in 9.9.

Figure 9.9: iostat 2 5

```
# iostat 2 5
   tty       cmdk0          sd0           nfs1          nfs2           cpu
 tin tout kps tps serv  kps tps serv  kps tps serv  kps tps serv   us sy wt id
   1  140  48   8    8    0   0    0    0   0    0    0   0    0    5  1  0 94
   0  117   0   0    0    0   0    0    0   0    0    0   0    0    1  0  0 99
   0   40   0   0    0    0   0    0    0   0    0    0   0    0    1  0  0 99
```

Chapter 10

Backup and Restore

Hard drives fail. People do silly things. Lightning strikes. These reasons and many more just drive home the fact that everyone should be backing up their data.

This chapter will give some information on different backup and restore utilities in Solaris as compared to Linux.

Topics Covered
File System Backup and Restore
Compression Tools
File System Snapshots

File System Backup and Restore

When backing up and restoring file systems on Linux you'll typically use the *dump* and *restore* commands.

Similarly, on Solaris you'll use the *ufsdump* and *ufsrestore* commands to backup UFS file systems. For ZFS file systems see Section 10, File System Snapshots.

There are many other backup tools that are used on Linux systems. Tar and cpio are examples of these tools that some use. They are also available on Solaris.

There are also a variety of open source backup tools, such as Amanda, that are popular in Linux circles. Just like most open source tools, Amanda can also be used in Solaris.

Sun also delivers The Sun StorageTek Enterprise Backup software and other associated storage management tools. Sun also resells Legato Networker and IBM Tivoli. There are many other third party backup software available on the Solaris platform.

Compression Tools

The three most common compression tools *bzip*, *gzip* and *zip* are all available on Solaris. You should confirm the options have not been changed.

File System Snapshots

Snapshots of the UFS file system can be taken using the *fssnap* utility. When using ZFS snapshots can be taken using the *zfs* command. A command similar to *zfs snapshot tank/home@monday* will produce a snapshot called *monday*. You can then take this snapshot data and write it to tape using:

```
zfs send tank/home@monday > /dev/rmt/0
```

Chapter 11

Troubleshooting

If Murphy is to believed, something will always go wrong. This chapter is here to help you out when that something happens. Following is a selection of issues and guidance to aid you in solving some issues you may encounter.

Topics Covered
Installation
System Startup
Core Files
Kernel Crash Dumps
Logs
Permissions and File Access Problems
Missing Commands
Printing
File Systems
Root Password Recovery
Network
Diagnostic and Debugging Tools

Installation

Installing from a USB CDROM drive

Installing Solaris 10 onto an x86 system from a bootable USB CDROM doesn't always complete successfully. Occasionally the system will lose track of the USB CDROM being used to install. If this happens the following should help rectify the problem.

1. During installation, select the Solaris Interactive Text (Console session) install type. Continue until the install drops to a shell with the message:

   ```
   ERROR: The disc you inserted is not a Solaris OS CD/DVD
   ```

2. Check, and remember, the list of available devices in /dev/dsk.
3. Unplug the USB CDROM and plug it back in after a few seconds.
4. Check for the newly recognized device (e.g. c1t0d0XXXX).
5. Mount the USB CDROM *mount -F hsfs /dev/dsk/c1t0d0p0 /cdrom*, remember it must be p0 (partition 0).
6. Run */sbin/install-solaris* to continue.

System Startup

If you run into issues with system startup there are a few places you can look for information. First, *svcs -x* will list any services that were not started, or that were offlined by the system. There will also be pointers to log files and other information to aid in correcting issues.

The next place to look is the */var/adm/message* file. This file contains all the log messages for the system. Failing that, verify that any scripts being executed from */etc/rc*.d* complete without issue.

Cannot mount boot archive

If you receive a `panic: cannot mount boot archive` message while booting then you've somehow lost your boot_archive. This can happen if you install multiple patches simultaneously that require rebuilding of the boot archive.

In order to correct this situation you'll need to boot the system into failsafe mode. Failsafe mode will normally prompt you to update the boot archive. If this is the case, reboot and you should be fine.

Otherwise, check if the original root partition is already mounted on */a*, if not, mount it to */a*. (e.g. *mount -F ufs /dev/dsk/c1d0s0 /a*). Once the partition is mounted issue */a/boot/solaris/bin/create_ramdisk -R /a* to rebuild the boot archive.

Reboot and the issue should be corrected.

Core Files

Core files are created when a system exits abnormally. Cores can be quite useful in determining the issue that caused the application to terminate. The *coreadm* utility allows you to modify the location, name and contents of core dump files produced on Solaris.

There are also several applications available to extract information from a core file. Some of these applications are listed in Table 11.1.

Table 11.1: Core file utilities

pflags	Print tracing flags.
pcred	Print credentials.
pldd	List dynamic libraries linked into the process.
pstack	Print a hex and symbolic stack trace for each LWP in the process.

Kernel Crash Dumps

If a kernel panic happens on a Linux machine an *oops* message will appear which can be decoded to determine the cause of the issue. There are no actual crash dump files produced by the Linux kernel.

If the Solaris kernel panics, a crash dump file will be written to */var/crash/hostname*. This crash dump file can be used in post-mortem analysis of why the system crashed.

Logs

When trouble occurs it is often a good idea to take a look at the log files as the first step to diagnosing the issue. Depending on the desired information there are a few different places to look. */var/adm/messages* contains the main system logs. */var/log/syslog* contains Sendmail logs, among others. */var/cron/log* will

contain information from the cron service. Finally, */var/lp/logs/lpsched* will contain the printer server logs.

There are also some application specific log files, for example:

- /var/samba/log
- /var/apache/log
- /var/apache2/log

Missing Commands

Commands may exist in different directories, may need to be installed, or may need to be ported if unavailable.

First, check the $PATH variable of your shell to verify all of the needed directories are present. The command could exist in one, or many, of the directories listed in Tables 1.2 or 1.3.

If the command isn't installed you can attempt to install a freeware package of the software. This could be retrieved from the Companion CD, sunfreeware.com, blastwave.org or installed from a source distribution.

If the tool doesn't exist on Solaris you may be required to port the original tool or write your own variation.

Printing

File Systems

Root Password Recovery

If you've managed to forget the root password for your Solaris box, or inherited a box without being told the password, it is possible to recover the box without having to re-install.

Boot from the Solaris install CD, then when it starts exit out of the install process. Mount the root disk to */a*, modify */a/etc/passwd* and */a/etc/shadow*. Reboot the system and the password should be changed.

Network

If you're having trouble with the network connections, Table 11.2 lists a few commands that may be of service.

Table 11.2: Useful networking commands

traceroute	To trace the route a packet takes to reach its destination.
/usr/sbin/ping	To test if remote hosts are reachable.
ifconfig	To view network interface configuration.
netstat	To show network status.
snoop	Similar to tcpdump.
tcpdump	Available on the companion CD.

Controlling NFS versions supported

In some circumstances, mounting a file system exported from a Linux system to a Solaris 10 system, the following error message may be encountered:

```
$ mount linux-nfs-server:/export /mnt
nfs mount: mount: /mnt: Not owner
```

This occurs because the Solaris 10 NFS client uses the NFSv4 protocol to mount the file system, and the Linux NFS client does not support NFSv4.

The solution to this problem is to change the default NFS client protocol that the Solaris system will use to the NFSv3 protocol. To do this, edit */etc/default/nfs* and add the line NFS_CLIENT_VERSMAX=3 to instruct Solaris to use the NFSv3 protocol until the Linux nfs server software is upgraded to support NFSv4.

If for some reason you want the Solaris nfs server not to support NFSv4, you can do this by modifying */etc/default/nfs* to indicate NFS_SERVER_VERSMAX=3 The Solaris system will now only support up to version 3 of the NFS protocol.

Diagnostic and Debugging Tools

DTrace is a powerful new tool available in Solaris 10 to make the process of diagnosing system issues easier. DTrace can probe into all aspects of the system including network, IO, function calls and when applications go on and off the CPU. See Section 2 for more information.

Several repositories exist for DTrace scripts. The OpenSolaris DTrace community[1] is a good starting point. The DTrace Toolkit is a collection of useful scripts that maybe of aid in your troubleshooting.

[1] http://opensolaris.org/os/community/dtrace/

Two other tools that are available are *truss* which can trace system calls made by applications and *apptrace* which is used to trace function calls made by an application.

Appendix A

Packages

The following is a list of the software packages that are bundled with, or supported on, Solaris 10. Any package that is listed *emphasized* is fully supported by Sun and Sun provides support the same as for Sun owned software. Otherwise, Sun provides existing patches and escalates new bugs to the developer community

Network Servers & Clients

Apache	Apache2	*bind*
Mozilla	ncftp	*ppp*
Samba	*sendmail*	*SER (SIP Proxy Server)*
Tomcat	wget	*wu-ftpd*
xntpd	*Zebra*	

Commands

a2ps	MySQL	bzip2
patch	footmatic print ppds	texinfo
ghostscript	*traceroute*	ghostscript fonts
Webmin	Gimp print drivers	gzip
GNU patch utility	GNU tar	GNU grep
less	ImageMagick	texi2html
IPMItool	mkisofs	Open Printing API
rpm2cpio.pl	System Management Agent	

Libraries

Glib	GTK+	JPEG
Libexpat	Libusb	Libxslt
PNG	Tcl/Tk	TIFF
XML2	XPM	zlib

Compilers & Tools

Binutils	gcc	Bison
gm4	Flex	gmake

Scripting Languages

Perl Python

Security Tools

Secure Shell tcp_wrappers

Shells

bash tcsh zsh

Applications

Networking

cups-1.1.20	ethereal 0.10.5	fetchmail 6.2.5
hpijs 1.6	lynx-2.8.4	mutt-1.4.2.1
nmap 3.5	nmh-1.0.4	Open LDAP 2.2.17
Open SLP 1.0.11	pine-4.61	procmail-3.22
rsync 2.6.3pre1	slm-0.9.6.2	snort-2.0.0
tcpdump-3.8.3		

Publishing

espgs-7.07.1 graphviz 1.10 groff-1.16.1
xpdf 3.0

Utilities

afio-2.4.6	amanda-2.4.4	cdrtools-2.01
cupsddk 1.0	diffutils-2.8.1	enscript-1.6.1
expect 5.39	file-4.10	fileutils-4.1
findutils-4.1.20	Foomatic filters 3.0.2	Foomatic-ppds 3.0.1
gcal-3.01	gettext-0.10.35	gimp-print-4.2.6
gkrellm 2.1.19	gnuplot 3.7.3	ispell-3.2.06
lxrun 0.9.6.1	mpack-1.5	mpage-2.5.1
mpg123-0.59r	mysql-jdbc-3.0.8	netpbm-10.3
plotutils-2.4.1	pnm2ppa-1.12	rpm-4.1
sane 1.0.12	screen 4.0.2	sgrep-1.92a
sh-utils-2.0a	sharutils-4.2.1	sudo 1.6.8p5
TeTex 2.0.2	textutils-2.0	tnef 1.1.3
top-3.5.1	uudeview-0.5.20	vorbis-1.0
wine 20041104	xpp-1.1	

Accessibility

brltty-3.3.1	emacspeak-18.0	emacspeak-ss-1.9.1
freetts-1.1.1	screenbrltty-4.02	unwindows-1.1.3
w3-4.0.47	yasr-0.6.4	

Editors

bluefish 0.12	emacs 21.3	gawk-3.0.6
joe-3.1	sed-3.02 (GNU)	vim-6.3
xemacs-21.4.15		

Development

Tools

autoconf 2.59	automake 1.8.3	binutils-2.15
cvs 1.11.17	ddd 3.3.8	gdb 6.2.1

Languages

bison-1.35	gcc-2.95.3	gcc-3.4.2
libtool 1.5.2	m4-1.4 (GNU)	MySQL python API 0.9.2
php-4.3.2	ruby-1.6.4	samp-1.0
tclX-8.2.0		

Libraries

aalib-1.2	berkley-db 1.85	berkley-db 4.2.52NC
curl-7.10.3	fltk-1.1.3	fnlib-0.5
GD Graphics library 2.0.15	guile-1.3.4	imlib-1.9.15
libexpt-1.95.7	libmpeg-1.3.1	libpcap-0.8.3
libsane 1.0.14	linungif-4.1.0	ncurses-5.2
Ogglib-1.0	Perl regex lib 4.5	qt-3.1.1

Desktop

Environment

kde-3.1.1a KOffice-1.2.1 XFce-3.8.16

System

Daemons

imap2002d (UW) proftpd 1.2.10rc1 squid 2.5.STABLE7

X

Applications

afterstep-1.8.8 fvwm2-2.4.3 WindowMaker-0.80.2

Window Managers

global-4.8	readline-4.2	make-3.80 (GNU)
slang-1.4.0	SDL-1.2.5	Xaw3d-1.5
asclock-1.0	xcpustate-2.5	ethereal-0.9.11
xdelta 1.1.3	gimp-1.2.1	xmcd 3.2.1
rxvt-2.7.10	xmms 1.2.10	stardic-1.3.1
xterm-196 (XFree86)	vnc-3.3.7	

Appendix B

Quick Reference Guide

The information in this chapter can be used as a quick reference when moving from Linux to Solaris. It will give information on the command changes and application differences.

Table B.1: Command Differences

Linux	*Solaris*
ps	/usr/bin/ps requires command line argument changes /usr/ucb/ps has compatible command line arguments but the output maybe different.
tcpdump	use snoop
awk	Use one of nawk, /usr/xpg4/bin/awk or gawk.
tar	Use /usr |opt/sfw/bin/gtar

Table B.2: Configuration Files

Linux	*Solaris*
/etc/fstab	/etc/vfstab
/etc/exports	/etc/dfs/dfstab
/etc/ntp.conf	/etc/inet/ntp.conf
/etc/aliases	/etc/mail/aliases
/etc/inetd.conf	inetadm (solaris 10)
/etc/xinetd.conf	inetadm (solaris 10) Solaris 9 - convert to /etc/inetd.conf
/etc/printcap	/etc/printers.conf

Table B.3: Kernel Drivers

Linux	*Solaris*
/etc/modules*	/etc/system
	/kernel/drv/*.conf

Table B.4: Kernel Configuration

Linux	*Solaris*
sysctl	/etc/system

Index

/etc/system, 43
/opt/sfw/bin, 4
/platform, 3
/proc, 2, 3
/usr/bin, 3
/usr/sfw/bin, 4
/usr/ucb, 3

adb, 43, 44
adddrv, 46
apptrace, 66
awk, 19, 20
    gawk, 20
    GNU awk, 19
    nawk, 20
    System V awk, 19
    XPG4 awk, 20

BART, 50
basename, 19
boot_archive, 62
bzip, 60

cat, 20
chmod, 23
chown, 20
chroot jails, 11
coreadm, 63
CUPS, 38

devfsadm, 46
df, 21, 56
DTrace, 14, 43
dtrace, 14, 44
du, 22, 56
dump, 60

edquota, 40
eeprom, 47
ext3, 2

fdisk, 40
filesystem(5), 3
Flash Archives, 32
FMRI, 8
format, 40
fsck, 16
fssnap, 60

getfacl, 25
Global Zone, 11
GNU, 19
GNU tar, 25
groupadd, 27, 38
gzip, 60

halt, 37

inetadm, 9, 51
inetconv, 9
inetd, 9
init, 36, 37
insmod, 43
iostat, 57

JFS, 2
Jumpstart, 30

kstat, 54

Least Privileges, 50
legacy_run, 8
Live Upgrade, 32
lofiadm, 40
lsmod, 43

metastat, 56

minicom, 47
modinfo, 43
modload, 43, 46
modprobe, 43
modunload, 43, 46
mpstat, 54

ndd, 43, 44
netstat, 55

patchadd, 34
patchrm, 34
pkgadd, 34
pkgrm, 34
Predictive Self Healing, 18
prstat, 57
ps, 22
psrinfo, 54

quot, 40
quota, 40

RBAC, 50
reboot, 37
reiser, 2
Resource Management, 11
restore, 60
rmmod, 43
rpm, 34
Run Levels, 36

setfacl, 23
setserial, 47
seyon, 47
Shells
    /bin/bash, 2
    /bin/sh, 2
smc, 38
SMF, 8, 18
smgroup, 38
smpatch, 34
smuser, 38
Solaris Containers, 11
Solaris Security Toolkit, 50
ssh, 43
Standards
    POSIX, 2
    SVID, 2
    XPG, 2
SunSolve, 34
svcadm, 8, 9, 51
svccfg, 9
svcs, 8, 9
sysctl, 43

tar, 25
telnet, 43
tip, 47
top, 57
truss, 66

UFS, 2
ufsdump, 60
ufsrestore, 60
up2date, 34
updatemanager, 34
useradd, 25, 38

Virtualization, 11
vmstat, 54
vold, 46

XFS, 2

ZFS, 2, 16, 41
zfs, 60
zip, 60
zoneadm, 12
zonecfg, 12
Zones, 11
    Sparse, 12
    Whole Root, 12