Solaris System Administration Series

Solaris™ 10 System Administration Essentials

Solaris System Engineers

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Sun Microsystems, Inc., has intellectual property rights relating to implementations of the technology described in this publication. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents, foreign patents, or pending applications. Sun, Sun Microsystems, the Sun logo, J2ME, J2EE, Solaris, Java, Javadoc, Java Card, NetBeans, and all Sun and Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC., MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales, international@pearsoned.com

Visit us on the Web: informit.com/ph

Library of Congress Cataloging-in-Publication Data

Solaris 10 system administration essentials / Solaris system engineers.
p. cm.
Includes index.
ISBN 978-0-13-700009-8 (pbk. : alk. paper)
1. Electronic data processing—Management. 2. Systems software. 3. Solaris (Computer file) I. Sun Microsystems.
QA76.9.M3S65 2009
005.4'3—dc22

2009034498

Copyright © 2010 Sun Microsystems, Inc.
4150 Network Circle, Santa Clara, California 95054 U.S.A.
All rights reserved.

Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

ISBN-13: 978-0-13-700009-8
ISBN-10: 0-13-700009-X
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, November 2009

Preface

Solaris™ 10 System Administration Essentials

Solaris™ 10 System Administration Essentials is the centerpiece of the new series on Solaris system administration. It covers all of the breakthrough features of the Solaris 10 operating system in one place. Other books in the series, such as *Solaris™ 10 Security Essentials* and *Solaris™ 10 ZFS Essentials*, cover specific features and aspects of the Solaris OS in detail.

Solaris™ 10 System Administration Essentials is the most comprehensive book about Solaris 10 on the market. It covers the significant features introduced with the initial release of Solaris 10 and the features, like ZFS, introduced in subsequent updates.

The Solaris OS has a long history of innovation. The Solaris 10 OS is a watershed release that includes features such as:

- **Zones/Containers,** which provide application isolation and facilitate server consolidation
- **ZFS,** the file system that provides a new approach to managing your data with an easy administration interface
- The **Fault Management Architecture,** which automates fault detection and resolution

- The **Service Management Facility,** a unified model for services and service management on every Solaris system
- **Dynamic Tracing (DTrace),** for troubleshooting OS and application problems on production systems in real time

The Solaris 10 OS fully supports 32-bit and 64-bit x86 platforms as well as the SPARC architecture.

This book is the work of the engineers, architects, and writers who conceptualized the services, wrote the procedures, and coded the rich set of Solaris features. These authors bring a wide range of industry and academic experience to the business of creating and deploying operating systems. These are the people who know Solaris 10 best. They have collaborated to write a book that speaks to readers who want to learn Solaris or who want to use Solaris for the first time in their company's or their own environment. Readers do not have to be experienced Solaris users or operating system developers to take advantage of this book.

The book's key topics include:

- Installing, booting, and shutting down a system
- Managing packages and patches (software updates)
- Controlling system processes
- Managing disks and devices
- Managing users
- Configuring networks
- Using printing services

Books in the Solaris System Administration Series

Solaris™ 10 Security Essentials

Solaris™ 10 Security Essentials describes how to make Solaris installations secure and configure the operating system to the particular needs of an environment, whether the systems are on the edge of the Internet or running a data center. It does so in a straightforward way that makes a seemingly arcane subject accessible to system administrators at all levels.

Solaris™ 10 Security Essentials begins with two stories that highlight the evolution of security in UNIX systems and the particular strengths that Sun Microsystems has added to the Solaris operating system that make it the best choice for meeting the present-day challenges to robust and secure computing.

Solaris™ 10 ZFS Essentials

Solaris™ 10 ZFS Essentials presents the revolutionary Zettabyte File System introduced in Solaris 10. It is a file system that is elegant in its simplicity and the ease with which it allows system administrators to manage data and storage.

ZFS is an all-purpose file system that is built on top of a pool of storage devices. File systems that are created from a storage pool share space with the other file systems in the pool. Administrators do not have to allocate storage space based on the intended size of a file system because file systems grow automatically within the space that is allocated to the storage pool. When new storage devices are added, all file systems in the pool can immediately use the additional space.

Intended Audience

The books in the Solaris System Administration Series can benefit anyone who wants to learn more about the Solaris 10 operating system. They are written to be particularly accessible to system administrators who are new to Solaris, and people who are perhaps already serving as administrators in companies running Linux, Windows, and/or other UNIX systems.

If you are not presently a practicing system administrator but want to become one, then this series, starting with the *Solaris™ 10 System Administration Essentials*, provides an excellent introduction. In fact, most of the examples used in the books are suited to or can be adapted to small learning environments like a home setup. Even before you venture into corporate system administration or deploy Solaris 10 in your existing IT installation, these books will help you experiment in a small test environment.

OpenSolaris

In June 2005, Sun Microsystems introduced OpenSolaris, a fully functional Solaris operating system release built from open source. While the books in this series focus on Solaris 10, they often incorporate aspects of OpenSolaris. Now that Solaris has been open-sourced, its evolution has accelerated even beyond its normally rapid pace. The authors of this series have often found it interesting to introduce features or nuances that are new in OpenSolaris. At the same time, many of the enhancements introduced into OpenSolaris are finding their way into Solaris 10. Whether you are learning Solaris 10 or already have an eye on OpenSolaris, the books in this series are for you.

6

Managing System Processes

This chapter discusses all the basic concepts for managing system processes in the Solaris Operating System. It covers:

- *Conditions of a process in Solaris*
- *Different states of a process*
- *Process context information*
- *Different commands and utilities present for monitoring and controlling the system processes in Solaris*
- *Process Manager utility for monitoring, controlling, and scheduling the system processes in Solaris*

6.1 Overview

The process is one of the fundamental abstractions of Unix. In Unix, every object is represented as either a file or a process. With the introduction of the `/proc` structure, there has been an effort to represent even processes as files.

A process is an instance of a running program or a program in execution. It can be any task that has an address space, executes its own piece of code, and has a unique process ID (PID). A process can create another process called a child process. Any process that creates the child process is called the parent process. This creation of new processes from existing parent processes is called forking (after the C function

called `fork()`). Most processes in the system are created by fork system calls. The `fork` system call causes the current process to be split into two processes: a parent process and a child process. The child process continues to execute on the CPU until it completes. On completion, the child process returns to the system any resources that it used during its execution. While the child process is running, the parent process either waits for the child process to complete or continues to execute. If the parent process continues to execute, it periodically checks for the completion of the child process.

Running multiple processes has an impact on system performance because the processes consume system resources, such as memory and processor time, and some processes may even cause the system to hang. Managing processes becomes important in a multiuser environment such as Solaris. Managing processes involves monitoring the processes, finding the resource usage, finding the parent processes that have created child processes, assigning priority for processes, scheduling processes, and terminating processes.

From a system administrator perspective there are three broad categories of tasks associated with the management of the systems processes:

- Monitoring the processes
  - Viewing the PID, UID, and PPID
  - Viewing the priority of the process
  - Viewing the resource usage (in terms of memory and processor utilization)
  - Viewing the state of the process, etc.
- Controlling the processes
  - Using signals
  - Assigning the priority to the processes
- Scheduling the processes

Note

Throughout this chapter we use an imaginary process, namely `proc_exp`, having `1234` as its process id (PID), with the following command line:

```
# proc_exp arg1 arg2 arg3
```

where `arg1`, `arg2`, and `arg3` represent process arguments.

6.1.1 State of a Process

A process undergoes many changes during its lifetime. For example, if a parent process waits for the child process to complete execution, the parent process puts itself in sleep state. Such a change from one state to another state is known as context switching. During its lifetime a process can exist in any of these four states: *Ready* or *Runnable, Running, Sleep*, and *Zombie*. A runnable process is ready to execute whenever CPU time is available. It has acquired all the resources it needs and is just waiting for the CPU to become available. If the process is in the Run state, it means that the process is running on the CPU. In the Sleep state, the process waits for a child process to complete or waits for a resource to become available. Zombie is the phase in which the child process terminates, but its entry is not removed from the process table until the parent process acknowledges the death of the child process by executing `wait()` or `waitpid()` system call. In this case, the child process is said to be in a Zombie state. Zombie processes are also called as *defunct* processes.

6.1.2 Process Context

Solaris is a multitasking, multiprocessing operating system, in which a number of programs run at the same time. A program can be made up of many processes. A process is a part of a program running in its own address space. This means that many users can be active on the system at the same time, running many processes simultaneously. But only one process is active per processor at any given time while the other processes wait in a job queue. Because each process takes its turn running in very short time slices (much less than a second each), multitasking operating systems give the illusion that multiple processes are running at the same time.

Each time a process is removed from access to the processor, sufficient information on its current operating state must be stored such that when it is again scheduled to run on the processor it can resume its operation from an identical position. This operational state data is known as its *context* and the act of removing the process's thread of execution from the processor (and replacing it with another) is known as a process switch or *context switch*.

The *context* of a process includes the following operational state data:

- Register set image
  - Program counter: address of the next instruction
  - Stack pointer: address of the last element on the stack

  - Processor status word: information about system state, with bits devoted to things like execution modes, interrupt priority levels, overflow bits, carry bits, etc.
  - Memory management registers: Mapping of the address translation tables of the process
  - Floating point unit registers
- User address space
  - Program text, data, user stack, shared memory regions, etc.
- Control information
  - u-area (user area), proc structure, kernel stack, address translation maps
- Credentials
  - User and group IDs (real and effective)
- Environment variables
  - Strings of the form variable = value

The **u area** includes the following:

- Process control block (PCB)
- Pointer to the proc structure
- Real/effective UID/GID
- Information regarding the current system call
- Signal handlers
- Memory management information (text, data, stack sizes)
- Table of open file descriptors
- Pointers to the current directory vnode and the controlling terminal vnode
- CPU usage statistics
- Resource limitations (disk quotas, etc.)

The **proc structure** includes the following:

- Identification: process ID and session ID
- Kernel address map location
- Current process state
- Pointers linking the process to a scheduler queue or sleep queue
- Pointers linking the process to lists of active, free, or zombie processes.
- Pointers keeping the structure in a hash queue based on PID

- Sleep channel (if the process is blocked)
- Scheduling priority
- Signal handling information
- Memory management information
- Flags
- Information on the relationship of this process and other processes

All of the information needed to keep track of a process when switching is kept in a data package called a *process control block (PCB)*. The process control block typically contains:

- Process ID (PID)
- Pointers to the locations in the program and its data where processing last occurred
- Register contents
- States of various flags and switches
- Memory information
- A list of files opened by the process
- The priority of the process
- The status of all I/O devices needed by the process

The new process is moved to the CPU by copying the PCB information into the appropriate locations (e.g., the program counter is loaded with the address of the next instruction to execute).

6.2 Monitoring the Processes

In Solaris, you can monitor processes that are currently executing on a system by using one of the commands listed in Table 6.1.

Table 6.1 Commands to Monitor the Processes

Command	Description
`ps`	Print status and information about active processes
`pgrep`	Find the process id (PID) of a process
`prstat`	View overall process statistics (similar to Linux top command)
`preap`	Reap zombie processes

continues

Table 6.1 Commands to Monitor the Processes (*continued*)

Command	Description
`pstop`	Temporarily freeze a process
`prun`	Continue a process that was stopped by `pstop` command
`pwait`	Wait for a process to finish
`pwdx`	List working directory for a process
`pargs`	Print the arguments and environment variables of a process
`pfiles`	Print the list of file descriptors associated with the process
`pldd`	List dynamic libraries associated with process (similar to `ldd` for executable)
`ptree`	Print a process ancestry tree
`pstack`	Get stack back trace of a process for debugging purposes
`truss`	Trace system calls and signals for a process
`svcs`	With the `-p` option, this command will list processes associated with each service instance. For more details, refer to Section 2.2, "Service Management Facility," in Chapter 2, "Boot, Service Management, and Shutdown."

Now let us examine each of the commands from Table 6.1 in more detail.

6.2.1 Process Status: `ps`

The `ps` command can be used to view the processes running on the system. Without options, `ps` prints information about processes that have the same effective user ID and the same controlling terminal as the invoker of `ps` command. The output contains only the process ID (PID), terminal identifier (TTY), cumulative execution time (TIME), and the command name (CMD). The output of the `ps` command is as shown below.

```
# ps
   PID TTY          TIME CMD
 27014 syscon       0:00 sh
 27151 syscon       0:00 ps
 27018 syscon       0:00 bash
```

You can print more detailed and comprehensive information about the running processes using different options available for the `ps` command, as described in Table 6.2.

Table 6.2 `ps` Command Options

Option	Description
`-a`	Lists information about all the most frequently requested processes. Processes not associated with a terminal will not be listed.
`-e`	Lists information about every process on the system.
`-A`	Lists information for all processes. Identical to the -e option.
`-f`	Lists full information for all processes.
`-l`	Generates a long listing.
`-P`	Prints the number of the processor to which the process is bound, if any, under an additional column header PSR. This is a useful option on systems that have multiple processors.
`-u <username>`	Lists only process data for a particular user. In the listing, the numerical user ID is printed unless you give `-f` option, which prints the login name.

Following is an example of using the `ps` command to list every process in the system:

```
# ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Apr 09 ?       1:15 sched
    root     1     0  0   Apr 09 ?       0:01 /sbin/init
    root     2     0  0   Apr 09 ?       0:00 pageout
    root     3     0  0   Apr 09 ?       7:06 fsflush
    root     7     1  0   Apr 09 ?       0:03 /lib/svc/bin/svc.startd
    root     9     1  0   Apr 09 ?       0:22 /lib/svc/bin/svc.configd
    root   505   504  0   Apr 09 ?       0:03 /usr/lib/autofs/automountd
  daemon   336     1  0   Apr 09 ?       0:00 /usr/lib/nfs/lockd
    root   151     1  0   Apr 09 ?       0:00 /usr/lib/picl/picld
    root   382     1  0   Apr 09 ?       0:02 /usr/lib/inet/inetd start
    root   170     1  0   Apr 09 ?       0:00 devfsadmd
  daemon   302     1  0   Apr 09 ?       0:00 /usr/bin/rpcbind
    root   311     1  0   Apr 09 ?       0:00 /usr/lib/netsvc/yp/ypbind
    root   144     1  0   Apr 09 ?       0:08 /usr/sbin/nscd
    root   616     1  0   Apr 09 ?       0:02 /usr/sfw/sbin/snmpd
    root   381     1  0   Apr 09 ?       0:00 /usr/sbin/cron
    root   313     1  0   Apr 09 ?       0:00 /usr/sbin/keyserv
  daemon   142     1  0   Apr 09 ?       0:00 /usr/lib/crypto/kcfd
  daemon   312     1  0   Apr 09 ?       0:00 /usr/lib/nfs/statd
    root   123     1  0   Apr 09 ?       0:00 /usr/lib/sysevent/syseventd
    root   159     1  0   Apr 09 ?       0:00 /usr/lib/power/powerd
    root   383   350  0   Apr 09 ?       0:00 /usr/lib/saf/ttymon
    root   350     7  0   Apr 09 ?       0:00 /usr/lib/saf/sac -t 300
```

Table 6.3 lists and describes the different process attribute fields displayed with the `ps` command.

Table 6.3 Process Attribute Fields

Field	Description
F	Flags associated with the process.
S	The state of the process. Refer to Table 6.4 for a complete list of the process states.
UID	The user ID of the process owner.
PID	The process ID of each process. This value should be unique. Generally, PIDs are allocated lowest to highest, but they wrap at some point.
PPID	The parent process ID. This identifies the parent process that started the process. Using the PPID enables you to trace the sequence of process creation that took place.
PRI	The priority of the process. Without `-c` option, higher numbers mean lower priority. With `-c` option, higher numbers mean higher priority.
NI	The nice value, used in priority computation. This is not printed when `-c` option is used. A process's nice number contributes to its scheduling priority. Making a process nicer means lowering its priority.
ADDR	The memory address of the process.
SZ	The SIZE field. This is the total number of pages in the process. Page size may vary on different hardware platforms. To display the page size on your system, issue the `/usr/bin/pagesize` command.
WCHAN	The address of an event for which the process is sleeping. If the address is `-`, then the process is running.
STIME	The starting time of the process (in hours, minutes, and seconds).
TTY	The terminal assigned to your process.
TIME	The cumulative CPU time used by the process in minutes and seconds.
CMD	The command (truncated) that generated the process.

Table 6.4 lists the codes used to show the various process states by the S field of the `ps` command.

Table 6.4 Process States

Code	Process state	Description
O	Running	The process is running on a CPU.
S	Sleeping	The process is waiting for an event to complete.
R	Runnable	The process is in the run queue.

Table 6.4 Process States (*continued*)

Code	Process state	Description
Z	Zombie	The process was terminated and the parent is not waiting.
T	Traced	The process was stopped by a signal because the parent is tracing it.
W	Waiting	The process is waiting for CPU usage to drop to the CPU-caps enforced limits.

6.2.2 Grepping for Process: `pgrep`

The `pgrep` command examines the active processes on the system and reports the process IDs of the processes whose attributes match the criteria specified on the command line. It can be used to replace the combination of the `ps` and `grep` commands to get the PID of a process based on some known process attributes. Following is an example of using the `pgrep` command. It also shows how the `pgrep` command can be used to replace the combination of the `ps` and `grep` commands:

```
# ps -e|grep sh
     3      ?         0:22    fsflush
  1238      pts/2     0:00    bash
   606      ?         0:00    sshd
  1234      pts/2     0:00    sh
  1274      pts/3     0:00    bash
  1270      pts/3     0:00    sh
#
# pgrep sh
3
1238
606
1234
1274
1270
```

6.2.3 Process Statistics Summary: `prstat`

The `prstat` command iteratively examines all active processes on the system and reports overall statistics on screen. The interesting thing about the `prstat` command is that information remains on the screen and gets updated periodically. By default, the `prstat` command updates the information screen every five seconds. However, the user can specify the sampling interval of choice on the command line. This command is similar to the `top` command in Linux. The syntax for the `prstat` command is as follows:

```
prstat [options] <count> <interval>
```

Table 6.5 describes some of the main options. Table 6.6 describes different arguments for the `prstat` command.

Table 6.5 Options for the `prstat` Command

Option	Description
`-a`	Displays separate reports about processes and users at the same time.
`-c`	Continuously prints new reports below previous reports instead of overwriting them.
`-n <nproc>`	Restricts the number of output lines. The `<nproc>` argument specifies how many lines of process or LWP (Light Weight Process or thread) statistics are reported.
`-p <pidlist>`	Reports only on processes that have a PID in the given list.
`-s <key>`	Sorts output lines by key in descending order. The four possible keys include: `cpu time, size, rss,` and `pri`. You can use only one key at a time.
`-S <key>`	Sorts output lines by key in ascending order.
`-t`	Reports total usage summary for each user.
`-u <uidlist>`	Reports only on processes that have an effective user ID (EUID) in the given list.
`-U <uidlist>`	Reports only on processes that have a real UID in the given list.

Table 6.6 Arguments for the `prstat` Command

Argument	Description
`<count>`	Specifies the number of times that the statistics are repeated. By default, `prstat` reports statistics until a termination signal is received.
`<interval>`	Specifies the sampling interval in seconds; the default interval is 5 seconds.

Following is an example of using `prstat` command with the sampling interval as one second:

```
# prstat 1
  PID  USERNAME SIZE  RSS    STATE PRI NICE TIME    CPU  PROCESS/NLWP
 796  noaccess 183M  100M   sleep 59  0    0:00:23 0.1% java/32
 1347 root     3440K 2888K  cpu5  59  0    0:00:00 0.0% prstat/1
 606  root     3520K 1280K  sleep 59  0    0:00:00 0.0% sshd/1
 567  root     2152K 1292K  sleep 59  0    0:00:00 0.0% snmpdx/1

(output edited for brevity)

 369  root     2040K 1164K  sleep 59  0    0:00:00 0.0% ttymon/1
 399 daemon    2448K 1272K  sleep 59  0    0:00:00 0.0% nfsmapid/6
 9    root     11M   9900K  sleep 59  0    0:00:05 0.0% svc.configd/14
 7    root     13M   11M    sleep 59  0    0:00:01 0.0% svc.startd/14
Total: 58 processes, 211 lwps, load averages: 0.00, 0.00, 0.00
```

Table 6.7 describes the column headings and their meanings in a `prstat` report.

Table 6.7 Column Headings for the `prstat` Command

Argument	Description
PID	The unique process identification number of the process
USERNAME	Login name or UID of the owner of the process
SIZE	The total virtual memory size of the process in kilobytes (K), megabytes (M), or gigabytes (G)
RSS	The resident set size of the process in kilobytes (K), megabytes (M), or gigabytes (G)
STATE	The state of the process: * cpu - The process is running on the CPU. * sleep - The process is waiting for an event to complete. * run - The process is in the run queue. * zombie- The process has terminated, and the parent is not waiting. * stop - The process is stopped.
PRI	The priority of the process
NICE	The value used in priority computation
TIME	The cumulative execution time for the process
CPU	The percentage of recent CPU time used by the process
PROCESS	The name of the process
NLWP	The number of lightweight processes (LWPs) or threads in the process

6.2.4 Reap a Zombie Process: `preap`

You can use the `preap` command to clean up a defunct or a zombie process. A zombie process has not yet had its exit status reaped, or claimed, by its parent. These processes are generally harmless, but can consume system resources if they are numerous. You need to specify the PID of the zombie process to be reaped with the `preap` command as shown below:

```
# ps -efl|grep Z
 F S UID   PID PPID C PRI NI  ADDR  SZ  WCHAN STIME TTY TIME CMD
 0 Z root 810 809  0 0   -   -     0   -        -  ?   0:00   <defunct>
 0 Z root 755 754  0 0   -   -     0   -        -  ?   0:00   <defunct>
 0 Z root 756 753  0 0   -   -     0   -        -  ?   0:00   <defunct>
#
# preap 810
810: exited with status 0
#
# ps -efl|grep Z
 F S UID   PID PPID C PRI NI  ADDR  SZ  WCHAN STIME TTY TIME CMD
 0 Z root 755 754  0 0   -   -     0   -        -  ?   0:00 <defunct>
 0 Z root 756 753  0 0   -   -     0   -        -  ?   0:00 <defunct>
```

In this example, the preap command successfully removed the zombie process with PID 810. Otherwise, the only way to remove them is to reboot the system.

6.2.5 Temporarily Stop a Process: pstop

A process can be temporarily suspended with the pstop command. You need to specify the PID of the process to be suspended as shown below:

```
# pstop 1234
```

6.2.6 Resuming a Suspended Process: prun

A temporarily suspended process can be resumed and made runnable with the prun command as shown below:

```
# prun 1234
```

6.2.7 Wait for Process Completion: pwait

The pwait command blocks and waits for termination of a process as shown below:

```
# pwait 1234
(sleep...)
```

6.2.8 Process Working Directory: pwdx

The current working directory of a process can be displayed using the pwdx command as shown below:

```
# pwd
/tmp/exp
# sleep 200
(sleep...)
# pgrep sleep
1408
# pwdx 1408
1408:   /tmp/exp
#
```

6.2.9 Process Arguments: pargs

The pargs command can be used to print the arguments and environment variables associated with a process. The pargs command solves a problem of the ps command

being unable to display all the arguments that are passed to a process. The `ps` command, when used with `-f` option, prints the full command name and its arguments, up to a limit of 80 characters. If the limit is crossed then the command line is truncated.

With the `-e` option, `pargs` command can be used to display the environment variables that are associated with a process. Following is an example of using the `pargs` command:

```
# ps -ef|grep proc
    root  1234  1008   0 21:29:13 pts/9       0:00 /bin/sh ./proc_exp arg1 arg2 arg3
#
# pargs 1234
1234:   /bin/sh ./proc_exp arg1 arg2 arg3
argv[0]: /bin/sh
argv[1]: ./proc_exp
argv[2]: arg1
argv[3]: arg2
argv[4]: arg3
#
# pargs -e 1234
1234:   /bin/sh ./proc_exp arg1 arg2 arg3
envp[0]: HZ=100
envp[1]: TERM=vt100
envp[2]: SHELL=/sbin/sh
envp[3]: PATH=/usr/sbin:/usr/bin
envp[4]: MAIL=/var/mail/root
envp[5]: PWD=/
envp[6]: TZ=Asia/Calcutta
envp[7]: SHLVL=1
envp[8]: HOME=/
envp[9]: LOGNAME=root
envp[10]: _=./proc_exp
```

6.2.10 Process File Table: `pfiles`

A list of files open within a process can be displayed with the `pfiles` command as shown below:

```
# pfiles 1368
1368:   /usr/sbin/in.rlogind
  Current rlimit: 256 file descriptors
   0: S_IFCHR mode:0000 dev:285,0 ino:64224 uid:0 gid:0 rdev:0,0
      O_RDWR|O_NDELAY
   1: S_IFCHR mode:0000 dev:285,0 ino:64224 uid:0 gid:0 rdev:0,0
      O_RDWR|O_NDELAY
   2: S_IFCHR mode:0000 dev:285,0 ino:64224 uid:0 gid:0 rdev:0,0
      O_RDWR|O_NDELAY
   3: S_IFDOOR mode:0444 dev:288,0 ino:55 uid:0 gid:0 size:0
      O_RDONLY|O_LARGEFILE FD_CLOEXEC  door to nscd[156]
      /var/run/name_service_door
   4: S_IFCHR mode:0000 dev:279,0 ino:44078 uid:0 gid:0 rdev:23,4
      O_RDWR|O_NDELAY
      /devices/pseudo/clone@0:ptm
```

continues

```
   5: S_IFCHR mode:0000 dev:279,0 ino:29885 uid:0 gid:0 rdev:4,5
      O_RDWR
      /devices/pseudo/clone@0:logindmux
   6: S_IFCHR mode:0000 dev:279,0 ino:29884 uid:0 gid:0 rdev:4,6
      O_RDWR
      /devices/pseudo/clone@0:logindmux
```

This example lists the files open within the `in.rlogind` process, whose PID is 1368.

6.2.11 Process Libraries: `pldd`

A list of the libraries currently mapped into a process can be displayed with the `pldd` command. This is useful for verifying which version or path of a library is being dynamically linked into a process. Following is an example of using the `pldd` command:

```
# pldd 1368
1368:   /usr/sbin/in.rlogind
/lib/libc.so.1
/lib/libsocket.so.1
/lib/libnsl.so.1
/lib/libbsm.so.1
/lib/libmd.so.1
/lib/libsecdb.so.1
/lib/libcmd.so.1
```

6.2.12 Process Tree: `ptree`

When a Unix process forks or initiates a new process, the forking process is called a *parent process* and the forked process is called a *child process*. This parent-child relationship can be displayed with the `ptree` command. When the `ptree` command is executed for a PID, it prints the process ancestry tree, that is, all the parents and children for this process, with child processes indented from their respective parent processes as shown below:

```
# ptree 1733
397   /usr/lib/inet/inetd start
  1731  /usr/sbin/in.rlogind
    1733  -sh
      1737  bash
        1761  ptree 1733
```

6.2.13 Process Stack: `pstack`

The `pstack` command can be used to print the stack trace of a running process. In case of a multi-threaded process, the stack trace of all the threads within a process will be displayed by default as shown below:

```
# pstack 1234
1234   ./proc_exp arg1 arg2 arg3
-----------------  lwp# 1 / thread# 1  --------------------
 fef74077 nanosleep (8047e10, 8047e18)
 080509e7 main     (4, 8047e60, 8047e74) + af
 080508a2 ???????? (4, 8047f10, 8047f1b, 8047f20, 8047f25, 0)
-----------------  lwp# 2 / thread# 2  --------------------
 fef74077 nanosleep (feeaefb0, feeaefb8)
 08050af2 sub_b    (0) + 1a
 fef73a81 _thr_setup (feda0200) + 4e
 fef73d70 _lwp_start (feda0200, 0, 0, feeaeff8, fef73d70, feda0200)
-----------------  lwp# 3 / thread# 3  --------------------
 fef74077 nanosleep (fed9efa8, fed9efb0)
 08050ac2 sub_a    (2) + ba
 fef73a81 _thr_setup (feda0a00) + 4e
 fef73d70 _lwp_start (feda0a00, 0, 0, fed9eff8, fef73d70, feda0a00)
-----------------  lwp# 4 / thread# 4  --------------------
 fef74ad7 lwp_wait (5, fec9ff8c)
 fef70ce7 _thrp_join (5, fec9ffc4, fec9ffc0, 1) + 5a
 fef70e29 thr_join (5, fec9ffc4, fec9ffc0) + 20
 08050d0d sub_d    (2) + a5
 fef73a81 _thr_setup (feda1200) + 4e
 fef73d70 _lwp_start (feda1200, 0, 0, fec9fff8, fef73d70, feda1200)
-----------------  lwp# 5 / thread# 5  --------------------
 fef74ad7 lwp_wait (3, feba0f94)
 fef70ce7 _thrp_join (3, feba0fcc, feba0fc8, 1) + 5a
 fef70e29 thr_join (3, feba0fcc, feba0fc8) + 20
 08050deb sub_e    (0) + 33
 fef73a81 _thr_setup (feda1a00) + 4e
 fef73d70 _lwp_start (feda1a00, 0, 0, feba0ff8, fef73d70, feda1a00)
```

The `pstack` command can be very helpful for debugging the process/thread hang issues. You can also print a specific thread's stack trace by supplying the thread-id (thread#) to the `pstack` command as shown below:

```
# pstack 1234/4
1234:   ./proc_exp arg1 arg2 arg3
-----------------  lwp# 4 / thread# 4  --------------------
 fef74ad7 lwp_wait (5, fec9ff8c)
 fef70ce7 _thrp_join (5, fec9ffc4, fec9ffc0, 1) + 5a
 fef70e29 thr_join (5, fec9ffc4, fec9ffc0) + 20
 08050d0d sub_d    (2) + a5
 fef73a81 _thr_setup (feda1200) + 4e
 fef73d70 _lwp_start (feda1200, 0, 0, fec9fff8, fef73d70, feda1200)
```

6.2.14 Tracing Process: `truss`

One of the most useful commands, `truss`, can be used to trace the system calls and signals made or received by a new or existing process. When used with `-d` flag, `truss` command prints a time stamp on each line of the trace output as shown below:

```
# truss -d date
Base time stamp:  1239816100.2290  [ Wed Apr 15 22:51:40 IST 2009 ]
 0.0000 execve("/usr/bin/date", 0x08047E78, 0x08047E80)  argc = 1
 0.0015 resolvepath("/usr/lib/ld.so.1", "/lib/ld.so.1", 1023) = 12
 0.0015 resolvepath("/usr/bin/date", "/usr/bin/date", 1023) = 13
 0.0016 sysconfig(_CONFIG_PAGESIZE)                     = 4096
 0.0016 xstat(2, "/usr/bin/date", 0x08047C58)           = 0
 0.0017 open("/var/ld/ld.config", O_RDONLY)             Err#2 ENOENT
 0.0017 mmap(0x00000000, 4096, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_ANON,
-1, 0) = 0xFEFF0000
 0.0018 xstat(2, "/lib/libc.so.1", 0x08047488)          = 0
 0.0018 resolvepath("/lib/libc.so.1", "/lib/libc.so.1", 1023) = 14
 0.0019 open("/lib/libc.so.1", O_RDONLY)                = 3
 0.0020 mmap(0x00010000, 32768, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_ALIGN, 3, 0) =
0xFEFB0000
 0.0020 mmap(0x00010000, 876544, PROT_NONE, MAP_PRIVATE|MAP_NORESERVE|MAP_ANON|MAP_
ALIGN, -1, 0) = 0xFEED0000
 0.0020 mmap(0xFEED0000, 772221, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_TEXT,
3, 0) = 0xFEED0000
 0.0021 mmap(0xFEF9D000, 27239, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_INIT-
DATA, 3, 774144) = 0xFEF9D000
 0.0021 mmap(0xFEFA4000, 5392, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANON,
-1, 0) = 0xFEFA4000
 0.0021 munmap(0xFEF8D000, 65536)                       = 0
 0.0023 memcntl(0xFEED0000, 123472, MC_ADVISE, MADV_WILLNEED, 0, 0) = 0
 0.0023 close(3)                                        = 0
 0.0025 mmap(0x00010000, 24576, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_
ANON|MAP_ALIGN, -1, 0) = 0xFEF90000
 0.0026 munmap(0xFEFB0000, 32768)                       = 0
 0.0027 getcontext(0x08047A10)
 0.0027 getrlimit(RLIMIT_STACK, 0x08047A08)             = 0
 0.0027 getpid()                                        = 2532 [2531]
 0.0027 lwp_private(0, 1, 0xFEF92A00)                   = 0x000001C3
 0.0028 setustack(0xFEF92A60)
 0.0028 sysi86(SI86FPSTART, 0xFEFA4BC0, 0x0000133F, 0x00001F80) = 0x00000001
 0.0029 brk(0x08062ED0)                                 = 0
 0.0029 brk(0x08064ED0)                                 = 0
 0.0030 time()                                          = 1239816100
 0.0030 brk(0x08064ED0)                                 = 0
 0.0031 brk(0x08066ED0)                                 = 0
 0.0031 open("/usr/share/lib/zoneinfo/Asia/Calcutta", O_RDONLY) = 3
 0.0032 fstat64(3, 0x08047CC0)                          = 0
 0.0032 read(3, " T Z i f\0\0\0\0\0\0\0\0".., 109)      = 109
 0.0032 close(3)                                        = 0
 0.0033 ioctl(1, TCGETA, 0x08047CE4)                    = 0
 0.0034 fstat64(1, 0x08047C50)                          = 0
Wed Apr 15 22:51:40 IST 2009
 0.0034 write(1, " W e d   A p r   1 5   2".., 29)      = 29
 0.0035 _exit(0)
```

The `truss` command is very helpful in debugging the process hang and the core dump issues. It can also be used to see which system process is taking more time and what parameters are passed for each system call.

You can use `-p` flag to specify the PID of the process to be traced as shown below:

```
# truss -p 1234
/4:     lwp_wait(5, 0xFEC8EF8C)         (sleeping...)
/3:     nanosleep(0xFED8DFA8, 0xFED8DFB0) (sleeping...)
/2:     nanosleep(0xFEEAEFB0, 0xFEEAEFB8) (sleeping...)
/5:     lwp_wait(3, 0xFEB8FF94)         (sleeping...)
/1:     nanosleep(0x08047E10, 0x08047E18) (sleeping...)
/2:     nanosleep(0xFEEAEFB0, 0xFEEAEFB8)               = 0
/1:     nanosleep(0x08047E10, 0x08047E18)               = 0
/1:     write(1, " M a i n   T h r e a d  "., 23)      = 23
/1:     lwp_sigmask(SIG_SETMASK, 0xFFBFFEFF, 0x0000FFF7) = 0xFFBFFEFF [0x0000FFFF]
/1:     lwp_exit()
/2:     write(1, " B :   T h r e a d   e x"., 21)      = 21
/2:     lwp_sigmask(SIG_SETMASK, 0xFFBFFEFF, 0x0000FFF7) = 0xFFBFFEFF [0x0000FFFF]
/2:     lwp_exit()
/3:     nanosleep(0xFED8DFA8, 0xFED8DFB0)               = 0
/3:     lwp_sigmask(SIG_SETMASK, 0xFFBFFEFF, 0x0000FFF7) = 0xFFBFFEFF [0x0000FFFF]
/3:     lwp_exit()
/5:     lwp_wait(3, 0xFEB8FF94)                         = 0
/5:     write(1, " E :   A   t h r e a d  "., 48)      = 48
/5:     write(1, " E :   J o i n   B   t h"., 17)      = 17
/5:     lwp_wait(2, 0xFEB8FF94)                         = 0
/5:     write(1, " E :   B   t h r e a d  "., 48)      = 48
/5:     write(1, " E :   J o i n   C   t h"., 17)      = 17
/5:     lwp_wait(0, 0xFEB8FF94)                         = 0
/5:     write(1, " E :   C   t h r e a d  "., 47)      = 47
/5:     nanosleep(0xFEB8FFA8, 0xFEB8FFB0) (sleeping...)
/5:     nanosleep(0xFEB8FFA8, 0xFEB8FFB0)               = 0
/5:     write(1, " E :   T h r e a d   e x"., 21)      = 21
/5:     lwp_sigmask(SIG_SETMASK, 0xFFBFFEFF, 0x0000FFF7) = 0xFFBFFEFF [0x0000FFFF]
/5:     lwp_exit()
...
```

You can use the `-t` flag to specify the list of specific system calls you are interested in tracing. In the following example the user is interested in tracing only `pread` and `pwrite` system calls for the process having a PID of 2614:

```
# truss -tpread,pwrite -p 2614
pread(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9EC3400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9F03400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9F03400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9F43400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9F43400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0\0"..., 262144, 0xC9F83400) = 262144
```

continues

```
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xC9F83400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xC9FC3400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xC9FC3400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA003400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA003400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA043400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA043400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA083400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA083400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA0C3400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA0C3400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA103400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA103400) = 262144
pwrite(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA143400) = 262144
pread(6, "\0\0\0\0\0\0\0\0\0\0\0".., 262144, 0xCA143400) = 262144
...
```

See the man pages for each of these commands for additional details.

6.3 Controlling the Processes

Controlling the processes in Solaris includes clearing hung processes, terminating unwanted or misbehaving processes, changing the execution priority of a process, suspending a process, resuming a suspended process, and so on. Following are the different ways the process can be controlled in Solaris.

6.3.1 The `nice` and `renice` Commands

If you wish to run a CPU intensive process, then you must know about the `nice` value of a process and the `nice` command. The `nice` value of a process represents the priority of the process. Every process has a `nice` value in the range from 0 to 39, with 39 being the nicest. The higher the `nice` value, the lower the priority. By default, user processes start with a `nice` value of 20. You can see the current `nice` value of a process in the **NI** column of `ps` command listing.

The `nice` command can be used to alter the default priority of a process at the start time. Following is an example of how to start a process with lower priority:

```
# nice -n 5 proc_exp arg1 arg2 arg3
```

This command will start the process `proc_exp` with `nice` value 25, which will be higher than the `nice` value 20 of other running processes and hence `proc_exp` will have lower priority.

Following is an example to start a process with higher priority:

```
# nice -n -5 proc_exp arg1 arg2 arg3
```

This command will start the process `proc_exp` with `nice` value 15, which will be less than the `nice` value 20 of other running processes and hence `proc_exp` will have higher priority.

The `renice` command can be used to alter the `nice` value of running processes. If `proc_exp` having PID 1234 was started with its default `nice` value of 20, the following command will lower the priority of this process by increasing its `nice` value to 25.

```
# renice -n 5 1234
```

or

```
# renice -n 5 -p 1234
```

The following command will increase the priority of `proc_exp` by decreasing its nice value to 15.

```
# renice -n -5 1234
```

or

```
# renice -n -5 -p 1234
```

For more information, see the `nice`(1M) and `renice`(1M) man pages.

6.3.2 Signals

Solaris supports the concept of signals, which are software interrupts. Signals can be used for communication between processes. Signals can be synchronously generated by an error in an application, such as SIGFPE and SIGSEGV, but most of the signals are asynchronous. A signal notifies the receiving process about an event. The following are the different ways to send a signal to a process:

- When a user presses terminal keys, the terminal will generate a signal; for example, when the user breaks a program by pressing the CTRL + C key pair.
- Hardware exceptions can also generate signals; for example, division by 0 generates SIGFPE (Floating Point Error) signal and invalid memory reference generates the SIGSEGV (Segmentation Violation) signal.
- The operating system kernel can generate a signal to inform processes when something happens. For example, SIGPIPE (Pipe Error) signal will be generated when a process writes to a pipe that has been closed by the reader.

- Processes can send the signal to other processes by using the `kill`(2) system call. Every process can send a signal in its privilege limitations. To send a signal, its real or effective user id has to be matched with the receiver process. Superuser can send signals without any restrictions.

There is also a Solaris command called `kill` that can be used to send signals from the command line. To send a signal, your real or effective `user id` has to be matched with that of the receiver process.

Every signal has a unique signal name and a corresponding signal number. For every possible signal, the system defines a default disposition, or action to take when it occurs. There are four possible default dispositions:

- Ignore: Ignores the signal; no action taken
- Exit: Forces the process to exit
- Core: Forces the process to exit, and creates a core file
- Stop: Stops the process (pause a process)

Programmers can code their applications to respond in customized ways to most signals. These custom pieces of code are called signal handlers. For more information on signal handlers, see the `signal`(3) man page.

Two signals are unable to be redefined by a signal handler. They are SIGKILL and SIGSTOP. SIGKILL always forces the process to terminate (Exit) and SIGSTOP always pauses a running process (Stop). These two signals cannot be caught by a signal handler.

Several other key points about signals are listed below:

- When a signal occurs, it is said that the signal is *generated*.
- When an action is taken for a signal, this means the signal is *delivered*.
- If a signal is between generation and delivery, this means the signal is *pending*, as clearly shown in Figure 6.1.
- It is possible to *block* a signal for a process. If the process does not ignore the blocked signal, then the signal will be pending.
- A blocked signal can be generated more than once before the process unblocks the signal. The kernel can deliver the signal once or more. If it delivers signals more than once, then the signal is *queued*. If the signals are delivered only once, then it is not queued. If multiple copies of a signal are delivered to a process while that signal is blocked, normally only a single copy of that signal will be delivered to the process when the signal becomes unblocked.

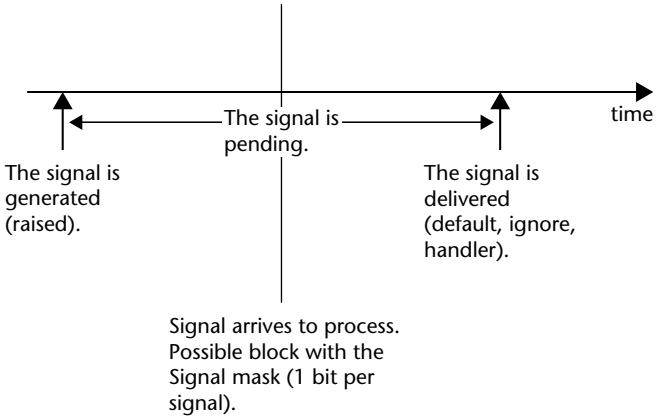

Figure 6.1 Signal States

- Each process has a signal mask. Signal masks define blocked signals for a process. It is just a bit array which includes one bit for each signal. If the bit is on, then that means the related signal will be blocked.

Note

The programmer can control (set or read) which signals are blocked (a blocked signal remains pending until the program unblocks that signal and the signal is delivered) with the `sigprocmask()` function. For more information, see the `sigprocmask` man page.

Table 6.8 provides the list of the most common signals an administrator is likely to use, along with a description and default action.

Table 6.8 Solaris Signals

Name	Number	Default Action	Description
SIGHUP	1	Exit	Hangup. Usually means that the controlling terminal has been disconnected.
SIGINT	2	Exit	Interrupt. User can generate this signal by pressing Ctrl+C.
SIGQUIT	3	Core	Quits the process. User can generate this signal by pressing Ctrl+\.
SIGILL	4	Core	Illegal instruction.

continues

Table 6.8 Solaris Signals (*continued*)

Name	Number	Default Action	Description
SIGTRAP	5	Core	Trace or breakpoint trap.
SIGABRT	6	Core	Abort.
SIGEMT	7	Core	Emulation trap.
SIGFPE	8	Core	Arithmetic exception. Informs the process of a floating point error like divide by zero.
SIGKILL	9	Exit	Kill. Forces the process to terminate. This is a sure kill. (Cannot be caught, blocked, or ignored).
SIGBUS	10	Core	Bus error.
SIGSEGV	11	Core	Segmentation fault. Usually generated when process tries to access an illegal address.
SIGSYS	12	Core	Bad system call. Usually generated when a bad argument is used in a system call.
SIGPIPE	13	Exit	Broken pipe. Generated when a process writes to a pipe that has been closed by the reader.
SIGALRM	14	Exit	Alarm clock. Generated by clock when alarm expires.
SIGTERM	15	Exit	Terminated. A gentle kill that gives the receiving process a chance to clean up.
SIGUSR1	16	Exit	User defined signal 1.
SIGUSR2	17	Exit	User defined signal 2.
SIGCHLD	18	Ignore	Child process status changed. For example, a child process has terminated or stopped.
SIGPWR	19	Ignore	Power fail or restart.
SIGWINCH	20	Ignore	Window size change.
SIGURG	21	Ignore	Urgent socket condition.
SIGPOLL	22	Exit	Pollable event occurred or Socket I/O possible.
SIGSTOP	23	Stop	Stop. Pauses a process. (Cannot be caught, blocked, or ignored).
SIGTSTP	24	Stop	Stop requested by user. User can generate this signal by pressing Ctrl+Z.
SIGCONT	25	Ignore	Continued. Stopped process has been continued.
SIGTTIN	26	Stop	Stopped—tty input.
SIGTTOU	27	Stop	Stopped—tty output.
SIGVTALRM	28	Exit	Virtual timer expired.
SIGPROF	29	Exit	Profiling timer expired.

Table 6.8 Solaris Signals (*continued*)

Name	Number	Default Action	Description
SIGXCPU	30	Core	CPU time limit exceeded.
SIGXFSZ	31	Core	File size limit exceeded.
SIGWAIT-ING	32	Ignore	Concurrency signal used by threads library.
SIGLWP	33	Ignore	Inter-LWP (Light Weight Processes) signal used by threads library.
SIGFREEZE	34	Ignore	Checkpoint suspend.
SIGTHAW	35	Ignore	Checkpoint resume.
SIGCANCEL	36	Ignore	Cancellation signal used by threads library.
SIGLOST	37	Ignore	Resource lost.
SIGRTMIN	38	Exit	Highest priority real time signal.
SIGRTMAX	45	Exit	Lowest priority real time signal.

Sometimes you might need to terminate or stop a process. For example, a process might be in an endless loop, it might be hung, or you might have started a long process that you want to stop before it has completed. You can send a signal to any such process by using the previously mentioned `kill` command, which has the following syntax:

```
kill [ - <signal>] <pid>
```

The `<pid>` is the process ID of the process for which the signal has to be sent, and `<signal>` is the signal number for any of the signal from Table 6.8. If you do not specify any value for `<signal>`, then by default, 15 (SIGTERM) is used as the signal number. If you use 9 (SIGKILL) for the `<signal>`, then the process terminates promptly.

However, be cautious when using signal number 9 to kill a process. It terminates the receiving process immediately. If the process is in middle of some critical operation, it might result in data corruption.

For example, if you kill a database process or an LDAP server process using signal number 9, then you might lose or corrupt data contained in the database. A good policy is to first always use the `kill` command without specifying any signal and wait for a few minutes to see whether the process terminates gently before you issue the `kill` command with -9 signal. Using the `kill` command without specifying the signal number sends SIGTERM (15) signal to the process with PID as

<pid> and thus the receiving process does the clean up job before terminating and does not result in data corruption.

As described earlier, the ps or pgrep command can be used to get the PID of any process in the system. In order to send SIGSTOP signal to process proc_exp, first you can determine the PID of proc_exp using the pgrep command as follows:

```
# pgrep proc_exp
1234
```

Now you can pass this PID to the kill command with signal number 23 (SIGSTOP) as follows:

```
# kill -23 1234
```

This will result in getting the process proc_exp paused.

There is another interesting Solaris command, pkill, which can be used to replace the pgrep and kill command combination. The pkill command works the same way as the kill command, but the only difference is, the pkill command accepts the process name as the last argument instead of PID. The syntax of the pkill command is as follows:

```
pkill  [ - <signal>] <process name>
```

The <process name> is the name of the process (command name) to which the signal has to be sent. You can use a single pkill command to send SIGSTOP signal to process proc_exp as follows:

```
# pkill -23 proc_name
```

For more information, see the kill(1M) and pkill(1M) man pages.

6.4 Process Manager

Both Solaris desktop environments–CDE and JDS–provide a GUI based Process Manager utility that can be used for monitoring and controlling systems processes. The advantage of using this GUI based Processor Manager is that you can

monitor and control system processes without any need to remember the complex commands and their syntax as discussed in this chapter so far. For example, instead of using the `ps` command with different options, you can invoke this Process Manager and it opens up showing all the system processes. You can sort the process list alphabetically, numerically, or based on any other field. You can use the filter text box to show only the processes that match the text typed in the filter box. You can search for a desired process by typing the relevant text in the find text box. You can terminate a process by highlighting it using the mouse pointer and then clicking kill.

In order to use the Process Manager utility, you need to log into the Desktop Environment of Solaris, either the Common Desktop Environment (CDE) or Java Desktop Environment (JDS). In CDE you can start the Processor manager by executing the `sdtprocess` command on the shell terminal, as shown below:

```
# sdtprocess &
```

or

```
# /usr/dt/bin/sdtprocess &
```

Alternatively, you can click Find Process on the Tools subpanel, as shown in Figure 6.2.

In JDS you can start the Process Manager either by executing the `sdtprocess` command or pressing Ctrl+Alt+Delete on the keyboard. The Process Manager window opens, as shown in Figure 6.3.

The Process Manager displays and provides access to processes that are running on a system. Table 6.9 describes the different fields displayed in the Process Manager window.

With the Process Manager, you can sort the processes on the system on the basis of any of the items in the given list. For example, if you click the CPU% column heading, the process list will be sorted and displayed on the basis of the CPU usage, as shown in Figure 6.4. The list updates every 30 seconds, but you can choose a value in the Sampling field of the Process manager to update the list as frequently as you like.

You can filter the processes that match the specified text. Type some text in the Filter text box in the Process Manager and press the Enter key. This displays the process entries that match the typed text. Figure 6.5 shows the processes containing `/usr/sbin` in their process entries.

Empty the Filter text box and press Enter to redisplay all the processes on the system.

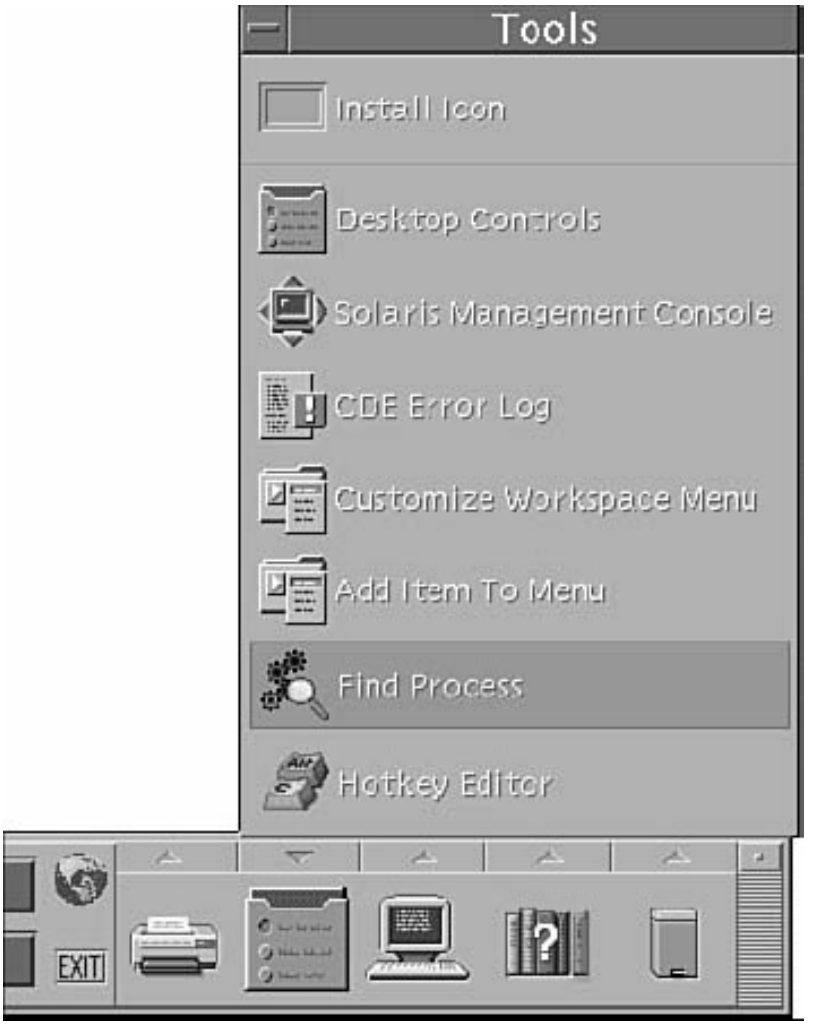

Figure 6.2 Tools Subpanel of CDE

Table 6.9 Fields in Process Manager Window

Column Heading	Description
ID	Process ID
Name	Name of the process
Owner	Login ID of the owner of the process
CPU%	Percentage of CPU time consumed

Table 6.9 Fields in Process Manager Window (*continued*)

Column Heading	Description
RAM	Physical memory or amount of RAM currently occupied by this process
Size	Total swap size in virtual memory
Started	Date when the process was started (or current time, if process was started today)
Parent	Parent process ID
Command	Actual Unix command (truncated) being executed

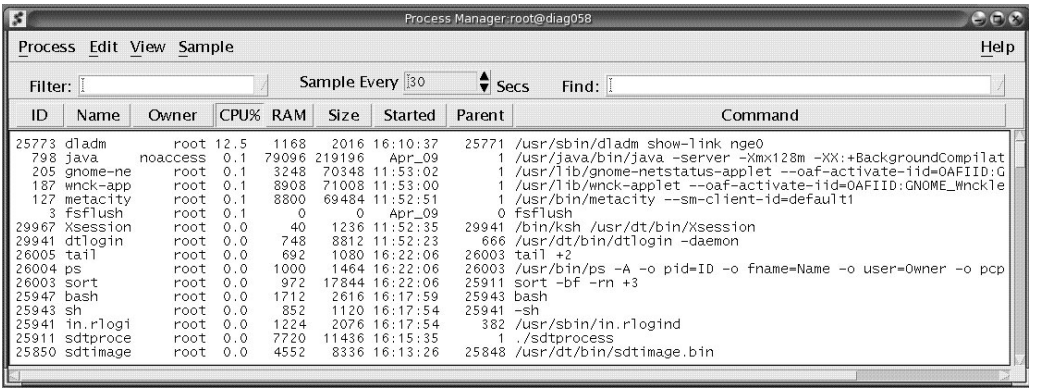

Figure 6.3 Process Manager Window

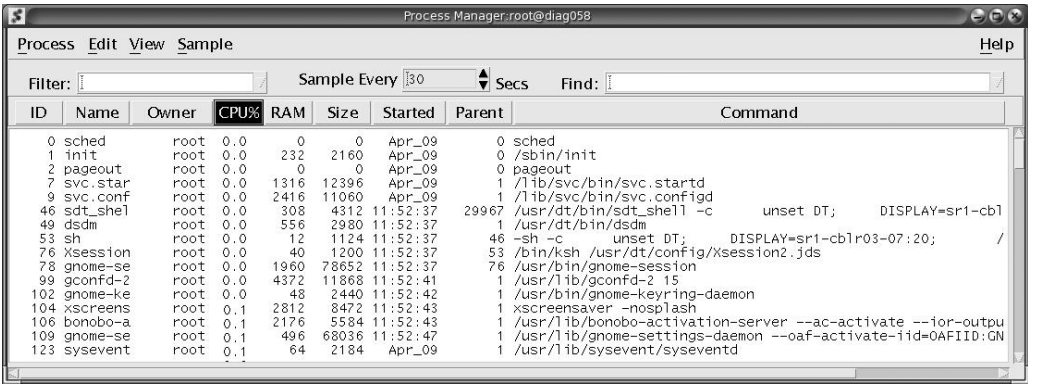

Figure 6.4 Process Manager Window Sorted by CPU%

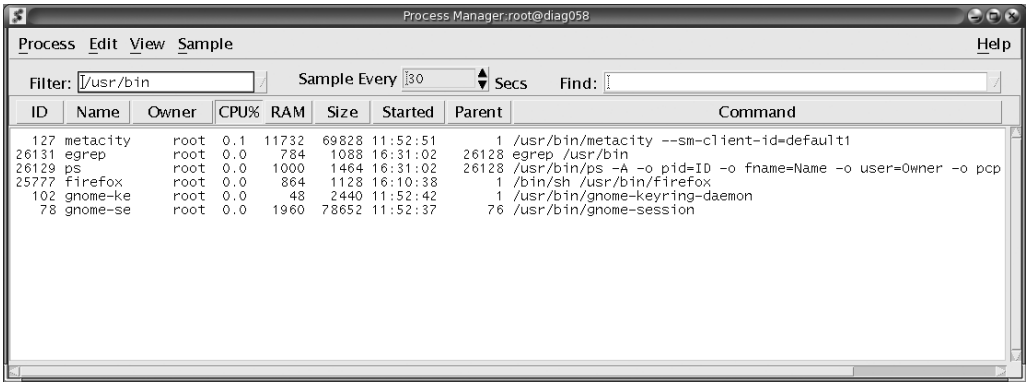

Figure 6.5 Process Manager Window after Specifying `/usr/bin` in the Filter Text Box

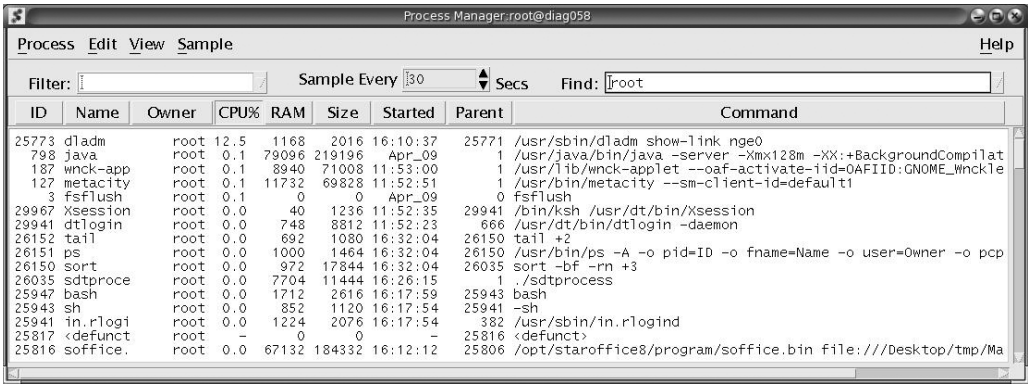

Figure 6.6 Process Manager Window after Specifying `root` in the Find Text Box

Using the **Find** box, processes containing the requested text string will be displayed in the Process Manager window. Type some text in the Find text box and press the Enter key. The processes containing the specified text will be displayed with the first occurrence of the specified text highlighted. This is shown in Figure 6.6.

Empty the Find text box and press Enter to redisplay all the processes on the system.

To kill a process, select or highlight the process from the listing and click the **Kill** option in the Process menu, shown at top of the window. This is shown in Figure 6.7. You also can use the Ctrl+C keyboard combination to kill the selected process or select the Kill option from the options that are available when you press the right mouse button. This will send SIGINT signal to the selected process.

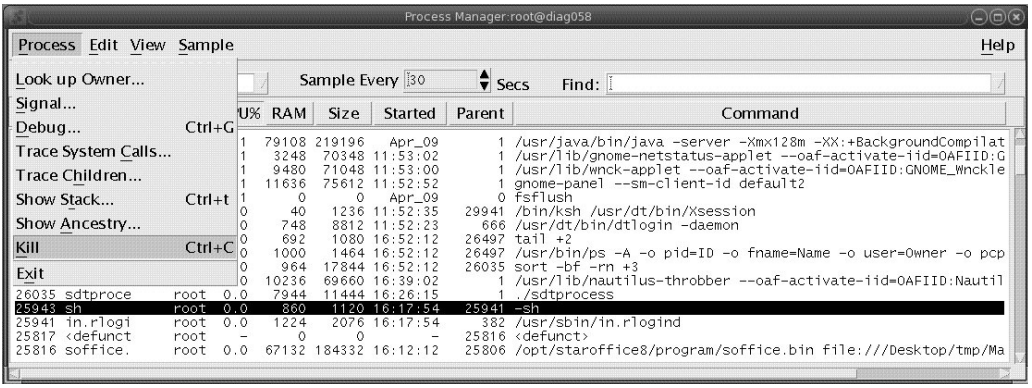

Figure 6.7 Process Manager Window with `kill` Selected

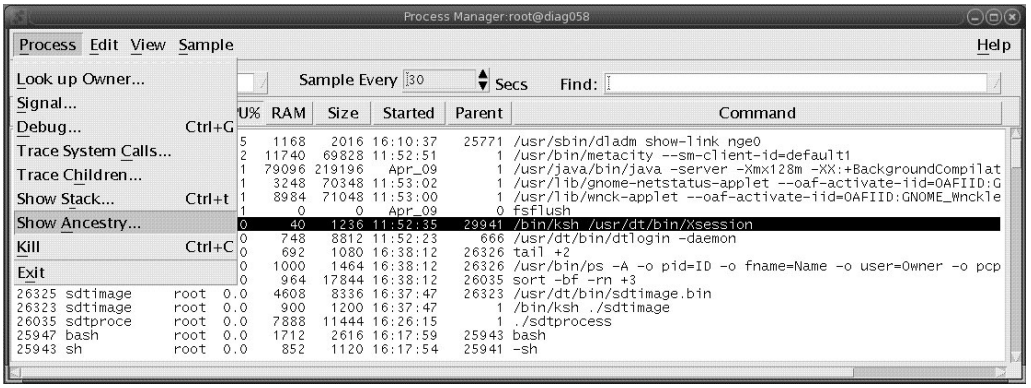

Figure 6.8 Process Manager Window with Show Ancestry Selected

You can also send signals of your choice to a process, similar to the signals sent from the command line using the `kill` command. For example, to send signal 9 (sure kill) for killing a process, select or highlight the process from the listing. Click the Process menu from the toolbar at the top of the Process Manager window and then click the **Signal** option. This will display a Signal window where you can specify 9 in the Signal text box and press Enter to kill the process.

Another interesting feature of the Process Manager utility is the capability to display the ancestry of a process. When a Unix process initiates one or more processes, they are called child processes or children. Child and parent processes have the same user ID. To view a process along with all its child processes, highlight the process in the Process manager window. Click the Process menu from the toolbar at the top of the Process Manager window and then click the **Show Ancestry** option, as shown in Figure 6.8. The Process Manager will display another window

containing the process tree for the specified process, as shown in Figure 6.9. Child processes are indented from the respective parent processes.

Figure 6.9 Show Ancestry Window

The command line equivalent to the Show Ancestry selection in the Process Manager is the `ptree` command, as described earlier in this chapter.

6.5 Scheduling Processes

From the user or system administrator's perspective, scheduling processes includes assigning priorities to the processes based on their importance and the need, executing a job at a time when the user will not be physically present at the system to manually start the job, distributing the job load over time, and executing a job repeatedly in a periodic fashion without manually having to start it each time.

Of the four tasks mentioned previously, the use of the `nice` and `renice` commands to assign priorities to the processes was described earlier in this chapter. Using these commands, you can increase the priority of a process that you want to complete faster. Similarly, if there is a long process taking

most of the CPU time and it is not important to get this process done fast, you can use these commands to reduce its priority so that other process will get to run more.

For the other three tasks, use the `crontab` utility and the `at` command described below.

6.5.1 `cron` Utility

The `cron` utility is a general Unix utility named after Chronos (meaning "time"), the ancient Greek god of time. It allows tasks to be automatically run in the background at regular intervals by the `cron` daemon. These tasks are often termed as `cron` jobs in Solaris.

Note

A daemon is a software process that runs in the background continuously and provides the service to the client upon request. For example, `named` is a daemon. When requested, it will provide DNS service. Other examples are:

- `sendmail` (to send/route email)
- `Apache/httpd` (web server)
- `syslogd` (the system logging daemon, responsible for monitoring and logging system events or sending them to users on the system)
- `vold` (the volume manager, a neat little daemon that manages the system CD-ROM and floppy. When media is inserted into either the CD-ROM or the floppy drive, `vold` goes to work and mounts the media automatically.)

Most of the daemons like those above and including `cron` are started at system boot up time and they remain active in the background until the system is shut down.

Crontab (CRON TABle) is a file that contains commands, one per line, that are read and executed by the `cron` daemon at the specified times. Each line or entry has six fields separated by space characters. The beginning of each line contains five date and time fields that tell the `cron` daemon when to execute the command. The sixth field is the full `pathname` of the program you want to run. These fields are described in Table 6.10.

The first five fields can also use any one of the following formats:

- A comma separated list of integers, like 1,2,4 to match one of the listed values.
- A range of integers separated by a dash, like 3-5, to match the values within the range.

Table 6.10 The `crontab` File

Field	Description	Values
1	Minute	0 to 59. A * in this field means every minute.
2	Hour	0 to 23. A * in this field means every hour.
3	Day of month	1 to 31. A * in this field means every day of the month.
4	Month	1 to 12. A * in this field means every month.
5	Day of week	0 to 6 (0 = Sunday). A * in this field means every day of the week.
6	Command	Enter the command to be run.

Note

- Each command within a `crontab` file must be on a single line, even if it is very long.
- Lines starting with # (pound sign) are treated as comment lines and are ignored.

The following are some examples of entries in the `crontab` file.

Example 6.1: Reminder

```
0 18 1,15 * * echo "Update your virus definitions" > /dev/console
```

This entry displays a reminder in the user's console window at 5.00 p.m. on 1st and 15th of every month to update the virus definitions.

Example 6.2: Removal of Temporary Files

```
30     17     *     *     *          rm /home/user_x/tmp/*
```

This entry removes the temporary files from `/home/user_x/tmp` each day at 5:30 p.m.

The `crontab` files are found in the `/var/spool/cron/crontabs` directory. All the `crontab` files are named after the user they are created by or created for. For example, a `crontab` file named `root` is supplied during software installation. Its contents include the following command lines:

```
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5/kprop_script ___slave_kdcs___
```

- The first command line instructs the system to run `logchecker` everyday at 3:10 a.m.
- The second command line orders the system to execute `nfsfind` on every Sunday at 3:15 a.m.
- The third command line runs each night at 3:30 a.m. and executes the `gsscred` command.
- The fourth command is commented out.

Other `crontab` files are named after the user accounts for which they are created, such as `puneet, scott, david,` or `vidya.` They also are located in the `/var/spool/cron/crontabs` directory.

When you create a `crontab` file, it is automatically placed in the `/var/spool/cron/crontabs` directory and given your user name. You can create or edit a `crontab` file for another user, or root, if you have superuser privileges.

6.5.1.1 Creating and Editing `crontab` Files

You can create a `crontab` file by using `crontab -e` command. This command invokes the text editor that has been set for your system environment. The default editor for your system environment is defined in the EDITOR environment variable. If this variable has not been set, the `crontab` command uses the default editor, `ed`, but you can choose an editor that you know well.

The following example shows how to determine if an editor has been defined, and how to set up `vi` as the default.

```
$ echo $EDITOR
$
$ EDITOR=vi
$ export EDITOR
```

If you are creating or editing a `crontab` file that belongs to `root` or another user, then you must become superuser or assume an equivalent role.

You do not need to become superuser to create or edit your own `crontab` file. The following are the steps to create a new or edit an existing `crontab` file:

1. Create a new `crontab` file, or edit an existing file.

   ```
   $ crontab -e [username]
   ```

The username specifies the name of the user's account for which you want to create or edit a crontab file. If you want to operate on your own crontab file then leave this option blank ($ crontab -e).

1. Add command lines to the crontab file.
   Follow the syntax described in Table 6.10.
3. Save the changes and exit the file. The crontab file will be placed in the /var/spool/cron/crontabs directory.
4. Verify your crontab file changes.

   # **crontab -l [username]**

   The contents of the crontab file for user <username> will be displayed.

   $ **crontab -l**

   The contents of your crontab file will be displayed.

6.5.1.2 Removing Existing crontab Files

You can use the crontab -r command to remove any existing crontab file. As noted previously, to remove a crontab file that belongs to root or another user, you must become superuser or assume an equivalent role.

crontab -r [username]

This will remove the crontab file for user <username>, if any.

$ **crontab -r**

This will remove your existing crontab file, if any.

6.5.1.3 Controlling Access to crontab

You can control access to crontab by modifying two files in the /etc/cron.d directory: cron.deny and cron.allow. These files permit only specified users to perform crontab tasks such as creating, editing, displaying, and removing their own crontab files. The cron.deny and cron.allow files consist of a list of user names, one per line. These access control files work together in the following manner:

- If cron.allow exists, only the users listed in this file can create, edit, display, and remove crontab files.
- If cron.allow doesn't exist, all users may submit crontab files, except for users listed in cron.deny.
- If neither cron.allow nor cron.deny exists, superuser privileges are required to run crontab.

Superuser privileges are required to edit or create `cron.deny` and `cron.allow`.

During the Solaris software installation process, a default `/etc/cron.d/cron.deny` file is provided. It contains the following entries:

- daemon
- bin
- nuucp
- listen
- nobody
- noaccess

None of the users listed in the `cron.deny` file can access `crontab` commands. The system administrator can edit this file to add other users who are denied access to the `crontab` command. No default `cron.allow` file is supplied. This means that, after the Solaris software installation, all users (except the ones listed in the default `cron.deny` file) can access `crontab`. If you create a `cron.allow` file, the only users who can access `crontab` commands are those whose names are specified in this `cront.allow` file.

For more information, see the `crontab` man page.

6.5.2 The `at` Command

Unlike the `cron` utility, which allows you to schedule a repetitive task to take place at any desired regular interval, the `at` command lets you specify a one-time action to take place at some desired time. For example, you might use `crontab` to perform a backup each morning at 4 a.m. and use the `at` command to remind yourself of a meeting later in the day.

6.5.2.1 Creating an `at` Job

To submit an `at` job, type at and then specify an execution time and a program to run, as shown in the following example:

```
# at 09:20am today
at> who > /tmp/log
at> <Press Control-d>
job   912687240.a       at    Thu    Jun  30    09:20:00
```

When you submit an `at` job, it is assigned a job identification number (912687240 in the case just presented), which becomes its filename along with the `.a` extension. The file is stored in the `/var/spool/cron/atjobs` directory.

The `cron` daemon controls the scheduling of `at` files similar to the way it does for `crontab` jobs.

The command syntax for `at` is shown here:

```
at [-m] <time> <date>
```

The `at` command syntax is described in Table 6.11.

Table 6.11 `at` Command Syntax

Option	Description
`-m`	Sends you mail after the job is completed.
`<time>`	The hour when you want to schedule the job. Add `am` or `pm` if you do not specify the hours according to a 24-hour clock (`midnight`, `noon`, and `now` are acceptable keywords). Minutes are optional.
`<date>`	The first three or more letters of a month, a day of the week, or the keywords `today` or `tomorrow`.

By default, users can create, display, and remove their own `at` job files. To access `at` files that belong to `root` or other users, you must have superuser privileges.

Following is an example of creating an `at` job:

```
$ at -m 2130
  at> rm /home/jonny/*.backup
  at> <Press Control-D>

  job  897355800.a  at  Thu Jun 30  21:30:00 2009
```

This shows the `at` job that user `jonny` created to remove his backup files at 9:30 p.m. He used the `-m` option so that he would receive an email message after the job is done.

6.5.2.2 Checking Jobs in Queue

To check your jobs that are waiting in the `at` queue, use the `atq` command. This command displays status information about the `at` jobs you created. You can also

use the atq command to verify that you have created an at job. The atq command confirms that at jobs have been submitted to the queue, as shown in the following example:

```
$ atq

Rank        Execution Date          Owner       Job   Queue  Job Name
1st         Jun 30, 2009 09:20      root  912687240.a  a     stdin
2nd         Jun 30, 2009 21:30      jonny 897355800.a  a     stdin
```

Another way to check an at job is to issue the at -1 command. This command shows the status information on all jobs submitted by a user, or for a particular job whose id is specified. The command syntax is as follows:

```
$ at -l [job-id]
```

The <job-id> is the identification number of the job whose status you want to display. If no <job-id> is specified, then status information on all jobs submitted by this particular user is displayed.

Following is an example using at -l command:

```
$ at -l
897543900.a    Sat Jul 14 23:45:00 2004
897355800.a    Thu Jul 12 19:30:00 2004
897732000.a    Tue Jul 17 04:00:00 2004
```

6.5.2.3 Removing Existing at Jobs

You can use the at -r command to remove any existing at job. Once again, to remove an at job that belongs to root or another user you must become superuser or assume an equivalent role. You do not need to become superuser or assume an equivalent role to remove your own at job.

```
# at -r [job-id]
```

This will remove the job with identification number job-id.

Verify that the at job has been removed by using the "at-1" or atq command to display the jobs remaining in the at queue. The job whose identification number you specified should not appear.

In the following example, a user wants to remove an at job that was scheduled to execute at 4 a.m. on July 17th. First, the user displays the at queue to locate the job identification number. Next, the user removes this job from the at queue

using the `at -r` command. Finally, the user verifies that this job has been removed from the queue using the `at -l` command:

```
$ at -l
897543900.a    Sat Jul 14 23:45:00 2003
897355800.a    Thu Jul 12 19:30:00 2003
897732000.a    Tue Jul 17 04:00:00 2003
$ at -r 897732000.a
$ at -l 897732000.a
at: 858142000.a: No such file or directory
```

6.5.2.4 Controlling Access to `at`

You can set up a file to control access to the `at` command, permitting only specified users to create, remove, or display queue information about their `at` jobs. The file that controls access to at is `/etc/cron.d/at.deny`. It consists of a list of user names, one per line.

The users listed in this file cannot access `at` commands. The default `at.deny` file, created during the Solaris OS software installation, contains the following user names:

- daemon
- bin
- smtp
- nuucp
- listen
- nobody
- noaccess

With superuser privileges, you can edit this file to add other user names whose `at` access you want to restrict.

For more information, see the `at`(1) man page.

Index

A

ABI (application binary interface), 47
`accept` command, print requests, 387
accumulation, patch, 97
ACLs (access control lists), 112–113
actions, database status and, 349
`add_drv` command, 255–256
`add_install_client`, 15
alternative service restarter, 43–44
Analyze menu, partition table, 215–217, 221–222
ancestry of process, Process Manager, 170–171
anonymous memory, 136
application binary interface (ABI), 47
application-independent shut down, 46
application-specific shut down, 46
architecture, Solaris print, 369–370
archive file systems, 124
`at` command
  checking jobs in queue, 176–177
  controlling access to, 178
  creating `at` job, 175–176
  defined, 175
  removing existing `at` job, 177–178
`atq` command, 176–177
attributes
  extended file, 115
  fields displayed with `ps` command, 148
  `rules` file, 15
`autofs` command, 101, 131
AUTOFS file system, 100
`auto_home` map, 132–133
`auto_master` file, 131
automatic log rotation, Fault Management, 185–186
automation tools, patch, 86–88
`automount` command, 131
`automountd` daemon, 131
automounting file systems, 130–133
  overview of, 130–131
  using direct maps, 132
  using indirect maps, 132–133
  using master maps, 131

B

backing up file systems
  defined, 107
  UFS, 107–108, 115–116
  before upgrading, 24
  ZFS, 120
backslash (\) character, 15

backup
of current functioning driver binaries, 251
and restore of SCF data, 45–46
begin scripts, 16–17
behavior
modifying boot, 36–37
run levels and corresponding system, 38
Berkeley Software Distribution Protocol (BSD), 372
BIOS, x86 Solaris installation, 9
block device interface, specifying disk subdirectory, 202
blocked signals, 160
blocks, disk, 199
boot archive, loading, 33–34
boot environment
defined, 63
limitations of activating, 78–79
patching for Solaris 10 8/07 release, 81–86
rebooting after applying/removing patches, 94–95
Solaris Live Upgrade, patching with, 63
Solaris Live Upgrade, planning, 77–78
Solaris Live Upgrade, process, 75–76
boot process, 33–39
bootloader, 33–34
grub extensions, 35–36
kernel, 34
modifying behavior, 36–37
run levels for OS, 37
troubleshooting, 37–39
user-mode programs, 34–35
booting
file system, 5
JumpStart server as boot server, 13–14
into single-user mode for patch installation, 93
SPARC installation, 23
x86 installation, 9, 24
from ZFS root pool, 26–27
zones, 331–332
bootloader, 33–34
bootparams file, 346
branded zones, 324
BrandZ (branded zone) framework, 324
broadcast method, NIS clients, 351
BSD (Berkeley Software Distribution Protocol), 372
bug fixes
patches delivering, 60
printing problems, 393
UFS logging, 114

C

cancel command, print requests, 388
caps, zone, 339
CDE (Common Desktop Environment), accessing Process Manager, 165
CDs
accessing information from, 134–135
adding device driver to installation image, 262–263
applying patch for Solaris 10 8/07 release, 80
installing software package from, 53
setting up spool directory from mounted, 55
x86 Solaris installation from, 10
certutil utility, 359
character strings, naming services/service instances, 39
check script, rules file, 20–21
checkinstall scripts, packages, 48
child processes
defined, 142
displaying process tree with ptree, 154
showing ancestry of, 170–171
and states, 143
chmod command, 311
class action scripts
package installation, 48
patches, 59
class drivers, 248–249
client hosts
configuring network domains, 282–283, 285–287
setting up network, 279
using naming service, 295
cloning zones, 333–334
column headings, prstat command, 151
COM1 (console ttya), x86 Solaris installation to, 10

COM2 (console ttyb), x86 Solaris installation to, 10
commented text, profile syntax, 17
Common Desktop Environment (CDE), accessing Process Manager, 165
Common UNIX Printing System (CUPS), IPP and, 373
components
  package, 48
  setting up network, 277–278
compression, ZFS supporting transparent, 124
compver file, packages, 48
.conf files, 192–193
configuration files
  fmd modules, 192–193
  TCP/IP networking stack, 276–277
configuration server, JumpStart Server as, 14
connections, printer
  overview of, 370
  troubleshooting network, 394
  troubleshooting no output, 392
context, process, 143–145
context switching, 143
contract services, SMF, 43
contract_id, SMF service implementation, 41
control files, packages, 48
copying packages to spool directory with pkgadd, 54–55
copy-on-write (COW) semantics, ZFS, 122–123
copyright file, packages, 48
COW (copy-on-write) semantics, ZFS, 122–123
cpio command, 116
CPU
  enabling compression on, 124
  temporarily modifying, 339
credentials, LDAP client
  configuring LDAP client, 358–359
  initializing LDAP client with proxy credentials, 362–363
  initializing using Per-User credentials, 357
  levels of, 356
  troubleshooting, 358–359
cron utility, 171–174
cron.allow file, 174–175
cron.deny file, 174–175
crontab files
  at command vs., 175
  controlling access to, 174–175
  creating, 173–174
  removing existing, 174
  scheduling processes, 171–173
CUPS (Common UNIX Printing System), IPP and, 373
cylinder group, UFS file system, 106
cylinders, disk, 199

D

daemons
  overview of, 171
  services implemented by, 42–43
  services not implemented by, 43
data corruption, patch management and, 62
datasets, ZFS, 117
DDI/DKI (device-driver interfaces/driver-kernel interfaces), 242
debugging
  printing problems, 393
  UFS logging, 114
dedicated dump device, 136
defect menu, partition table, 217
deferred activation patches, 89, 95
deleting
  packages, 18–19
  printer, 390–391
  zones, 336
delivered signals, 160
demo Kit, FMA, 193–194
depend file, packages, 48
dependencies
  patch, 96–98
  Service Management Facility (SMF), 41–42
  Solaris Live Upgrade, 77
/dev/console, 332
/dev namespace, 245–246, 325
devfs device file system, 243, 325
device-driver interfaces/driver-kernel interfaces (DDI/DKI), 242

Device Driver Utility, 237
device drivers. *See* drivers
device file system (`devfs`), 243, 325
device ID, PCI devices, 236–237
device interface, specifying disk subdirectory, 202
device numbers, 245
device tree, 243–244, 264–265
devices, 236–251
  administration commands, 270–272
  drivers and, 235–236
  installing and updating drivers. *See* drivers, installing and updating
  obtaining information about, 236–247
  reference information, 272–273
  searching for supported driver, 248–251
  when drivers hang or panic system, 265–270
  when it does not work, 236
  in zones, 325
`/devices` directory, 243–244, 245–246
`devsadm` command, 203
`df-k` command, UFS, 116
DHCP Configuration wizard, 285
DHCP server
  configuring server, 283
  configuring client host, 285–286
  setting up network, 283–284
  setting up PXE boot, 259–261
diagnosis engines, Fault Management, 180
direct maps, automount service, 132
directly attached printers, 370, 381–382
directories
  defined, 99
  home, exporting into non-global zone from global zone, 337
  initializing Native LDAP client, 365–366
  JumpStart, 14–15
  LDAP, 343
  NFS sharing, 127
  UFS, 105–106
directory services. *See* naming services
`disable` command, print requests, 387, 390
`disabled` subcommand, services, 40
disk controllers, 198–199
disk labels (VTOC or EFI)
  creating, 224–225
  defined, 199
  disk management using, 203–205
  printing using `fmthard`, 230–232
  printing using `prtvtoc`, 229–230
disk slices
  arrangements on multiple disks, 207–208
  creating, 224–225
  creating UFS file system on, 106
  displaying information, 223–224
  managing, 205–207
  running `newfs` command on, 107
disk space
  information about allocating, 374
  managing with quotas in UFS, 107–108
  recommendations for Software Groups, 4–5
  Solaris Live Upgrade planning, 77
  UFS logging consuming, 114
disks, 197–234
  accessing information from, 134–135
  checking UFS file system integrity, 110–112
  creating file system on disk, 228–229
  creating slices and labeling disk, 224–228
  creating UFS file systems on, 106–107
  determining if formatted, 218–219
  device drivers, 202–203
  device naming conventions, 200–202
  disk labels (VTOC or EFI), 203–205
  displaying partition table or slice information, 223–224
  expanding pool in ZFS by adding, 125–126
  formatting, 219–220
  hard disk drive, 197–198
  identifying defective sector with surface analysis, 221–222
  identifying on system, 218
  partition table. *See* partition table
  repairing defective sector, 222
  replacing in ZFS, 127
  slices. *See* disk slices
  terminology, 199
  when to use `patchadd`, 64
disks, useful commands
  `du`, 233–234
  `fmthard`, 230–232
  `fsck`, 232–233
  `prtvtoc`, 229–230
displaying faults, 182–184

distributed file systems, Solaris OS supporting, 100
dladm commands, 276, 303
DNS (Domain Name System)
initializing LDAP client using Per-User credentials, 357–358
NIS vs., 342–343
organizational use of, 344
overview of, 342
providing naming services with makefile, 291–292
setting up and configuring naming services, 350–351
Solaris installation with, 3
DNS namespace, 342
Domain Name System. *See* DNS (Domain Name System)
domain names, setting up network components, 278
domains
adding NIS users to NIS, 294–295
enabling name server to support multiple, 293
dot-dot file, 106
dot file, 106
downtime, patching with Solaris Live Upgrade and, 63
driver binary, installing manually, 253–256
drivers
adding device support to existing, 256
avoiding hard hangs, 254–255
avoiding recurring panic, 254
commands for managing, 270–272
defined, 199
device names, 245–246
device tree, 243–244
introduction to Solaris, 235–236
loading and testing new, 256
locating modules for, 243
module major and minor numbers, 245
obtaining information about, 236–237
obtaining information using CLI, 238–241
obtaining information using GUI, 237–238
online resources, 272
overview of, 202–203
registering new, 255
searching for supported, 248–251
Solaris kernel modules, 241–242
suggested reading, 272–273
using one that is for similar device, 250
version information, 246–247
when current driver does not work, 250
x86 Solaris installation, 11–12
drivers, installing and updating, 251–265
adding to net installation image. *See* installation image, adding device driver to
backing up current functioning binaries, 251
install time updates and, 252–253
manual binary installation, 253–256
package installations, 252
swapping disks, 263–265
dtrace command, 69
DTrace (Dynamic Tracing), 269, 339
du command, 233–234
dump device, 136
dumpadm command, 136
DVDs
accessing information from, 134–135
adding device driver to installation image, 262–263
applying patch for Solaris 10 8/07 release, 80
installing software package from, 53
installing Solaris 10 OS, 1–2
installing Solaris 10 OS with image, 1–2
setting up spool directory from, 55
x86 Solaris installation from, 10
dynamic routing, 295–296
Dynamic Tracing (DTrace), 269, 339

E

edquota command, UFS, 109–110
EFI (Extensible Firmware Interface) disk labels
creating disk labels, 224–225
displaying slice information, 224
overview of, 203–204
partition table for, 209
printing using fmthard, 230–232
printing using prtvtoc, 229–230
slice configuration for, 205–207

EIS (Enterprise Installation Standards), 88
enabled subcommand, services, 40
error detectors, Fault Management, 180
error log (errlog), Fault Manager
  automatic log rotation, 185–186
  defined, 184
  manual log rotation, 186–187
  viewing, 185
ESC key, x86 Solaris installation, 12
/etc/default/password file, 312
/etc/group file
  defined, 312
  overview of, 314
  verifying group/user account has been created, 317
/etc/hosts file, 354
/etc/inet/netmasks file, 346
/etc/mnttab file
  defined, 101
  determining file system type, 105
  unmounting file system and removing entry from, 102
/etc/netmasks file, 346
/etc/nsswitch.conf file
  configuring, 347
  DNS setup and configuration, 350–351
  example of, 348
  as naming service switch file, 347–349
  network database entries in, 344–345
  NIS setup and configuration, 351
  printer support in, 377–378
/etc/passwd file
  defined, 312
  overview of, 313–314
  verifying account is created, 316–317
/etc/resolv.conf file, DNS, 350
/etc/shadow file, 312, 314
/etc/syslog.conf file, 393
/etc/users_attr file, 314
/etc/vfstab file
  activating TMPFS file system, 138
  determining file system type, 104–105
  enabling UFS logging, 114
  file systems using, 103–104
  managing mounting of file systems, 101
  setting quotas in UFS, 109
  using loopback file system, 139
ethers database, 346
ETM (Event Transport Module), Fault Management, 192
Event Transport Module (ETM), Fault Management, 192
Events Registry, FMA, 194–196
exclusive-IP zones, 324–325
exporting, home directories into non-gobal zone, 337
extended file attributes, UFS, 115
Extensible Firmware Interface. *See* EFI (Extensible Firmware Interface) disk labels

F

failsafe boot, 39
failsafe entry, 33
fault log (fltlog), Fault Manager
  automatic log rotation, 185–186
  defined, 184
  manual log rotation, 186–187
  viewing, 185
Fault Management, 179–196
  directories, 193
  displaying faults, 182–184
  displaying message about device, 236
  downloadable Events Registry, 194–196
  downloadable FMA Demo Kit, 193–194
  Fault Notification messages, 181–182
  further information on, 196
  managing fmd and fmd modules, 188–193
  managing log files, 184–188
  overview of, 179–180
  repairing faults, 184
Fault Management Resource Identifiers. *See* FMRIs (Fault Management Resource Identifiers)
fault notification messages, 181–182
fdisk keyword, 19
fdisk menu, 214–215
fdisk partitions, 206
field-replaceable units (FRUs), Fault Management, 180, 183–184
fields, partition table, 208–209
file systems
  backing up before upgrading, 24–25
  configuring automatic NFS with sharing, 130

creating on disk, 228–229
determining type of, 104–105
managing users, groups and roles, 310
monitoring, 105
mounting, 100–102, 374
NFS. *See* NFS (Network File System)
overview of, 99
permissions for, 310–311
pseudo, 136–139
reference information, 139–140
removable-media, 133–136
services implemented by kernel for, 43
types supported by Solaris OS, 100
UFS. *See* UFS (UNIX file system)
unmounting, 102–103
using `/etc/vfstab` file, 103–104
ZFS. *See* ZFS file system
files
adding group and user to local, 315–316
defined, 99
listing process, 153–154
NFS server sharing, 127
user management and, 312–314
filtering processes, Process Manager, 165–166
Find text box, Process Manager, 169
`finish` script, packages or patches, 21–22
firewall gateways, as multihomed hosts, 288
`flar` (flash archive), swapping disks, 263–264
`fltlog`. *See* fault log (`fltlog`), Fault Manager
FMA (Fault Management Architecture) Demo Kit, 193–194
FMA (Fault Management Architecture) Events Registry, 194–196
`fmadm config` command, 189–191
`fmadm faulty` command, 182–184
`fmadm repair` command, 184
`fmadm rotate` command, 185–186
`fmadm rotate` logname command, 186–187
`fmadm unload` module-name command, 189
`fmd` (Fault Manager daemon)
configuration files, 192–193
defined, 180
loading and unloading modules, 189–191
managing Fault Management log files, 184–188
manual log rotation, 186–187
module descriptions, 189
rotation of log files, 185–186
tracking statistics, 191–192
`fmd.conf`, 192–193
`fmdump` command
displaying message about device problem, 236
examining historical log files, 188
viewing Fault Management log files, 185
`fmdump -v -u` UUID command, 183
FMRIs (Fault Management Resource Identifiers)
displaying faulty components in system, 183
naming service instances, 39
repairing faults, 184
SMF dependencies with, 41–42
`fmstat` command, 191–192
`fmthard` command, 230–232
forking process, 142, 154
`format -e` command, 203
`format` menu, partition tables, 211–213
`format` utility
creating disk slices and labels, 224–228
determining if disk is formatted, 218–219
displaying disk's partition table, 208–209, 211
displaying partition table or slice information, 223–224
formatting disks, 210–211, 219
identifying disks on x86-based system, 218
identifying/repairing defective sector, 221–222
free hog slice, 225
FRUs (field-replaceable units), Fault Management, 180, 183–184
`fsck` command
defined, 102
disk management, 232–233
UFS integrity, 111
`fsdb` command, 114
`fssnap` command, 115–116
`fsstat` command, 105
`fstyp` command, 104
`fuser` command, 102, 135

G

generated signals, 160
Genesis patches, 92
`getent` command, 301, 368
`getfacl` command, 112–113
GID (group ID), 312
global zones
  checking status of SMF in, 338
  configuring, 328
  defined, 321
  exporting home directories into non-global zone, 337
  overview of, 323–324
`group` database, 345
group ID (GID), 312
group name, 312
`groupadd` command, 312, 315–316
`groupdel` command, 312
`groupmod` command, 312
groups
  adding to local files, 315–316
  adding to NIS domain, 316–317
  managing, 314–317
  overview of, 310
grub
  bootloader, 33–34
  extensions, 35–36
  modifying boot behavior, 36–37
GRUB menu
  JumpStart installation/upgrade on x86, 23–25
  x86 Solaris installation, 9–10
GUI installer, 2–3, 11

H

`halt` command, application-independent shutdown, 47
halting, zones, 333
hangs. *See* hangs, system
hangs, system
  avoiding, 254–255
  device drivers causing, 266–268
  overview of, 265–266
hard disk drive, 197–198
hard limits, setting quotas, 108–109
hardware
  free patches for, 60, 73
  generating signals, 159
  troubleshooting no output (nothing prints), 392
Hardware Compatibility List (HCL), 249, 250
HCL (Hardware Compatibility List), 249, 250
heads, disk, 198
High Sierra file system (HSFS), 100, 136
home directory
  defined, 312
  exporting non-gobal zone from global zone, 337
hosts file, 345
`hosts.byname` file, NIS maps, 352–353
HSFS (High Sierra file system), 100, 136

I

IDR (Interim Diagnostic Relief) patches, 91
`ifconfig` command, 276, 281, 303–304
Image Packaging System (IPS), 252
image_directory, JumpStart installation/upgrade on x86, 24
implicit repairs, Fault Manager, 184, 193
importing, service manifests into SCF, 46
incompatibility, patch, 97–98
indirect maps, automount service, 132–133
`inetd` alternative restarter, 43–44
`init` command
  application-dependent shutdown, 47
  applying patch for Solaris 10 8/07 release, 85
  changing run level while OS is running, 37
  when to patch in single-user mode, 93
Initialize a Client Using Proxy Credentials, 362–363
`inittab` command, 34
`in.named` daemon, DNS servers, 342
inodes, UFS
  checking UFS integrity, 110–112
  defined, 105
  in multiterabyte UFS, 115
  using quotas to manage, 108
install server, JumpStart server as, 14
Install Time Update. *See* ITU (Install Time Update) package

installation
device driver. *See* drivers, installing and updating
JumpStart. *See* JumpStart installation
packages, 48
patches, 73–74
proactively patching during, 71–73
Solaris 10 OS. *See* Solaris 10 OS installation
zones, 331–332
installation image, adding device driver to, 256–263
adding driver, 261–262
to CD/DVD, 262–263
network PXE boot installation, 257
overview of, 256–257
setting up DHCP server for PXE boot, 259–261
setting up JumpStart PXE boot server, 257–258
installer, managing software packages, 49
`install_type` profile keyword, 17
instance names, disks, 200
instance numbers, devices, 245
integrity, checking UFS, 110–112
interactive configuration, 287
interactive patches, 92
Interim Diagnostic Relief (IDR) patches, 91
Interim Security Relief (ISR) patches, 91
Internet Printing Protocol (IPP), 373, 378
Internet Protocol (IP) layer, 3, 324
IP addresses
configuring NAT box for a domain, 299–300
configuring router for a domain, 280–282, 283–284
configuring DHCP server, 285
DNS services translating hostnames into, 342
Solaris installation with, 3
IP filter daemon, configuring NAT, 300–301
IP (Internet Protocol) layer, 3, 324
IPP (Internet Printing Protocol), 373, 378
IPS (Image Packaging System), 252
`isainfo` command, 246–247
ISR (Interim Security Relief) patches, 91
ITU (Install Time Update) package
installing only on OpenSolaris systems, 252
overview of, 252–253
x86 Solaris installation, 11

J

Java Desktop Environment (JDS), accessing Process Manager, 165
JDS (Java Desktop Environment), accessing Process Manager, 165
journaling file systems, 113–114
JumpStart installation
creating profile server, 14–21
custom, 22–25
defined, 2
deploying Solaris OS over network from, 257–258
overview of, 13
setting up JumpStart server, 13–14
upgrading and installing patches, 71–73
using `Finish` scripts, 21–22

K

`kdmconfig` utility, 11–12
Kerberos authentication, 3, 357–358
kernel
boot process and, 34
generating signals, 159
implementing file system services, 43
managing device drivers, 202–203
modifying boot behavior, 36–37
process contract with, 42–43
starting user-mode programs, 34–35
troubleshooting boot process, 39
kernel modules
device drivers, 241–242
installing. *See* drivers, installing and updating
location of, 243
Solaris device tree, 243–244
system and application interaction with, 242
kernel patches
deferred activation patching, 95
defined, 88
`kill` command, 160, 163–164
Kill option, Process Manager, 169–170
`kill(2)` system call, 160
`kmdb` debugger, 39

L

ladap_client_cred, LDAP, 356
ladap_client_file, LDAP, 356
language, configuring x86 system, 12
LDAP API listings, 368
LDAP (Lightweight Directory Access Protocol)
  API entry listings, 368
  configured switch file, 348–349
  configuring client, 359–362
  initializing client manually, 363–364
  initializing client using per-user credentials, 357–359
  initializing client using profiles, 362
  initializing client using proxy credentials, 362–363
  initializing native client, 364–368
  overview of, 343, 356
  Solaris installation with, 3
  troubleshooting client configuration, 364
  troubleshooting name service information, 368
  uninitializing client, 364
ldapclient command
  configuring LDAP client, 360
  defined, 356
  initializing LDAP client, 362
  initializing LDAP client manually, 363
  initializing Native LDAP client, 364–368
  modifying manual LDAP client configuration, 363–364
  setting up LDAP client, 356
leaf-node drivers, 243
libraries, listing process, 154
Lightweight Directory Access Protocol. *See* LDAP (Lightweight Directory Access Protocol)
limitpriv property, 338
line printer print service. *See* LP (line printer) print service
listening service, IPP, 378
listing system zones, 336
loading, new driver, 256
local file systems
  mounting, 101
  types supported by Solaris OS, 100
  using as naming service, 344
local printer, 370
lockfs command, UFS transaction log, 114
LOFS (loopback file system)
  deferred activation patching, 89
  defined, 100
  exporting home directories into non-gobal zone from global zone with, 337
  using, 139
log files
  Fault Management, 184–188
  locating installation, 8
  reactive patch management, 69–70
  SMF services, 40–41
  Solaris Live Upgrade planning, 77
  UFS, 113–114
logadm command, 185–187
logfile, SMF services, 40–41
logical device names, disks, 200–201
logical device tree, devfs, 243–244
login
  changing root from user to role, 318
  directory, 312
  name, 312
  shell, 312
  SMF, 35
  using zlogin command for zones, 332–333
loopback file system. *See* LOFS (loopback file system)
LP (line printer) print service
  overview of, 371–372
  troubleshooting no output, 392
  using commands, 385–391
lpadmin command, 390–391
LPD protocol, print client/print server communications, 372
lpmove command, 389–390
lpr.debug log, 393
lpstat command
  listing request IDs for printer, 389, 391
  overview of, 386–387
  verifying enabled printer, 387
ls command, 113, 311
luactivate command
  activating boot environment, 78–79
  applying patch for Solaris 10 8/07, 84–85
  Solaris Live Upgrade process, 75–76
lucompare command, 78
lucreate command
  activating boot environment, 78
  applying patch for Solaris 10 8/07, 81–83

creating new boot environment, 64, 94
Solaris Live Upgrade, 27, 75–76
ludelete command, 75–76
lustatus command, 78, 85
luumount command, 78
luupgrade command, 64, 70, 75–76

M

maintenance state, transferring service to, 42–43
major numbers, devices, 245
make command, 355
makefile file, 291–292, 355
man command, 102
man pages, accessing, 102
Management Information Base (MIB) plug-in, 182
manual configuration, LDAP client, 356
maps
automount service master, 131
NIS, 352–355
master maps, automount service, 131
master servers, NIS, 343, 351, 355
memory
diagnosing problems in driver code, 269
patch management for corruption of, 62
printer setup, 374
Solaris install requirements, 3
memory caps, zones, 339
MIB (Management Information Base) plug-in, 182
migrating zones, 334–336
milestone none, 39
miniroot image, defined, 261
minor numbers, devices, 245
mirroring, ZFS using, 121
modify command, format utility, 225–228
modinfo command, 246–247
modload command, 247
modules, fmd, 189–193
monitoring
file systems, 105
network performance, 302–307
SMF service manager, 42–43
mount command
activating boot environment, 78
information about, 102
managing media without using removable media, 135
mounting file systems, 101–102
setting up spool directory from remote package server, 55
mount point, 99–101
mount -v command, 101–102, 104
mountall command, 101
mounting file systems
with automounting, 130–133
defined, 99
information on, 374
NFS, 129
overview of, 100–102
zones, 328
mounting removable media, 134
moving zones, 333
mpstat command, 269
multihomed host, 287–295
adding NIS users to NIS domain, 294–295
configuring, 288–289
configuring naming services, 289–290
enabling name server to support multiple domains, 293
overview of, 287–288
preparing client host to use naming service, 295
preparing makefile script to provide naming services, 291–292
preparing source files for naming services, 290–291
setting up name server, 293–294
setting up network, 279
multiterabyte UFS, 115

N

N2L (NIS to LDAP) service, 344
name servers
enabling to support multiple domains, 293
preparing makefile script to provide naming services, 292–293
namespace
DNS, 342
NIS, 343
setting up client of LDAP, 356

naming conventions
devices, 245–246
disk devices, 200–201
packages, 49
services and service instances, 39
naming service cache daemon (nscd), 342
naming services, 341–368
adding printers in, 377
configuring root role into environment using, 319
configuring system for, 289–290
defined, 310
DNS, overview of, 342
DNS setup and configuration, 350–351
LDAP, overview of, 343
LDAP setup and configuration. *See* LDAP (Lightweight Directory Access Protocol)
naming service cache daemon (nscd), 342
network database sources, 344–347
NIS, overview of, 342–343
NIS setup and configuration, 350–355
organizational use of, 343–344
overview of, 341
preparing client host, 295
preparing makefile script, 291–292
preparing source files, 290–291
printer support in switch file, 377–378
setting up name server, 293–294
setting up network components, 278
switch file, 347–349
troubleshooting, 368
troubleshooting printer network connections, 394
using, 341–342
NAT (Network Address Translation)
configuring in NAT box, 300–301
configuring NAT box, 299–300
Native LDAP client, initializing, 364–368
netgroups, LDAP client, 361–362
netmasks database, 346
netstat command, performance, 304–305
network-attached printers, 370, 381–384
network database sources, for naming services, 344–347
network domain, configuring, 283–287
Network File System. *See* NFS (Network File System)
Network Information Service. *See* NIS (Network Information Service)
network interface cards (NICs), TCP/IP, 275–276
network interfaces, zones, 324–325
network protocols, print administration, 372–373
networks, 275–307
configuring as superuser, 277
information on, 307
monitoring performance, 302–307
Solaris installation with settings for, 3–4
TCP/IP stack for, 275–277
networks database file, 347
networks, setting up
components, 277–278
configuring domains, 279–283, 283–287, 298–301
configuring multihomed host, 287–295
setting up static routing, 295–298
testing configuration, 301–302
newfs command
creating file system on disk, 228–229
creating multiterabyte UFS, 115
creating UFS, 107
newzone command, 328–329
next_state, SMF services, 40–41
nexus drivers, 243
NFS (Network File System), 127–133
automatic file system sharing, 130
automounting, 130–133
clients and servers, 127
as distributed file system, 100
finding available, 128–129
mounting, 129
as multihomed host, 288
overview of, 127–128
unmounting, 129
using extended file attributes, 115
NFSv4, Solaris 10 OS supporting, 128
nice command, processes, 158–159, 170–171
NICs (network interface cards), TCP/IP, 275–276
NIS namespace, 343
NIS (Network Information Service)
adding groups and users to domain, 316–317

adding NIS users to NIS domain, 294–295
configuring system for naming services, 289–290
enabling name server to support multiple domains, 293
naming services setup and configuration, 351–355
organizational use of, 343–344
overview of, 342–343
preparing source files, 290–291
setting up name server, 293–294
setting up network, 279
Solaris installation with, 3
NIS to LDAP (N2L) service, 344
`nisswitch.ldap` file, 366–367
non-global zones
adding dedicated device to, 337
altering privileges in, 338
checking status of SMF services in, 338
defined, 321
exporting home directories in global zones into, 337
IP types for networked, 324
overview of, 323–324
patch metadata for, 95
proactive patching on, 66–68
reactive patching on, 70
required elements to create, 327–328
using DTrace program in, 339
non-system (secondary) disks, 208
nonstandard patches, 91
`nscd` (naming service cache daemon), 342
`nslookup` command, 302
`nsswitch` file, 357–358
`nsswitch.conf` file, 342
numbering, patch, 61

O

object permissions, file systems, 310–311
obsolescence, patch, 97
`offline` state, services
defined, 42
dependencies and, 40
troubleshooting services, 44–45
online resources
adding printers in naming service, 377
disk space allocation, 374
file systems, 139–140
file systems, backing up, 24
file systems, mounting, 374
`fmd.conf` parameters, 193
installing Solaris patches, 73
Interim Diagnostic Relief patches, 91
Interim Security Relief patches, 91
Internet Printing Protocol, 373
JumpStart installation, 72
LDAP API listings, 368
LP print commands, 386–387
naming services configuration, 289
networking, 307
NFS version for server or client, selecting different, 128
non-global zones, 66
patch rejuvenation, 90
patches for installed non-global zones, 67
PostScript Printer Definition File Manager, 371
preconfiguration, 287
printer definitions, 380
removable media, 134
role management, 319
Solaris 10 documentation, 27
Solaris for x86 Device Support list, 249
Solaris Hardware Compatibility List, 249
Solaris Live Upgrade and upgrade planning, 75, 85
Solaris passwords, 312
submitting RFE for driver to support your device, 250
Sun Alerts registration, 71
Sun Device Detection Tool, 237–238
Sun's patch testing, 89
UFS file system, 106
user IDs, 312
ZFS file system, 118
`online` state, services, 44
Open Boot PROM, 267
Open Systems Interconnection (OSI) model, TCP/IP, 275–276
OpenLDAP, 364–368
OSI (Open Systems Interconnection) model, TCP/IP, 275–276

P

Package Manager tool, 252
package metaclusters, 4
package objects, 48
packages, software, 47–57
  adding or deleting using profiles, 18–19
  adding with `finish` script, 22
  applying patch for Solaris 10 8/07 release, 79–80
  content of, 48
  copying to spool directory with `pkgadd`, 54–56
  defined, 47
  installing from mounted CD, 53
  installing from remote package server, 53
  installing ITUs
  installing with Genesis patches, 92
  installing with `pkgadd`, 50–52, 252
  managing, 47
  migrating zone to new system, 333–335
  naming conventions, 49
  overview of, 47
  preparing to configure Native LDAP client, 367
  removing with `pkgrm`, 50–51, 56–57
  tools for managing, 49–50
  in zone environment, 325–326
panic, system, 254, 268–270
PAPI (Print Application Programming Interface), 370
parent processes
  defined, 142
  displaying process tree with `ptree`, 154
  showing ancestry of, 170–171
  states and, 143
`pargs` command, 146, 152–153
Partition menu, 213–214, 223–224
partition table, 208–217
  Analyze menu, 215–217
  Defect menu, 217
  defined, 208
  displaying information, 223–224
  EFI-labeled disk example, 209
  `fdisk` menu on x86 systems, 214–215
  fields, 208–209
  `format` menu and command descriptions, 211–213
  `format` utility, 210–211
  Partition menu, 213–214
  VTOC-labeled disk example, 209
partitioning disks, 224–228
`passwd` command, 294
`passwd` database, 345
passwords, 312
Patch Check Advanced (PCA), 87
patch ID, 61
`patchadd` command
  applying patch for Solaris 10 8/07 with, 80, 84
  identifying problems before applying patches, 68
  patch metadata for non-global zones, 95
  patching systems with non-global zones installed, 67–68
  reactive patching using, 70
  standard patch installation not using, 91
  when Solaris Live Upgrade is inappropriate, 62–63
  when to apply patches with, 64–66
patches, 59–98
  adding with `finish` script, 22
  applying latest revision of, 62
  automation tools, 86–88
  content of, 60
  dependencies, 96–98
  metadata for non-global zones, 95
  migrating zone to new system, 333–335
  numbering, 61
  to OpenLDAP server for `ldapclient init` utility, 365
  overview of, 61
  proactive patch management. *See* proactive patch management
  reactive patch management, 68–70
  README special instructions, 92–95
  security management with, 70–71
  software management with, 59
  with Solaris Live Upgrade. *See* Solaris Live Upgrade
  types of, 88–92
  understanding, 59–60
  in zone environment, 325–326
Patches and Updates page, 73, 89
`patchrm` command, 60
`path_to_inst` instance database, 203
PCA (Patch Check Advanced), 87

PCB (Process Control Block), 145
PCFS (PC file system), 100, 135
PCI ID Repository Web site, 237
pending signals, 160
Per-User credentials
  configuring LDAP client, 358
  initializing LDAP client, 357–358
  troubleshooting, 358
  troubleshooting LDAP client configuration, 358–359
performance
  device drivers degrading, 268–270
  improve with UFS logging, 114
  measuring with ZFS, 124–125
  monitoring network, 302–307
  patches enhancing, 60
permissions, file system object, 310–311
pfiles command, 146, 153–154
pfinstall command, 19–21
pgrep command, 145, 149
physical device names, disks, 200
physical device tree, devfs, 243–244
ping command, 301
pkgadd command
  copying frequently installed packages to spool directory, 54–56
  defined, 50
  installing package from remote server, 53
  installing packages, 51–52
  overview of, 50–51
pkgchk command, 50, 52
pkginfo command, 50, 55
pkginfo file, 48
pkgmap file, 48
pkgparam command, 50
pkgparam command, 326
pkgrm command
  copying frequently installed packages to spool directory, 54
  defined, 50
  overview of, 50–51
  removing software packages, 56–57
pkgtrans command, 50
pkill command, 164
platters, disk, 198–199
pldd command, 146, 154
point patches, 90
poweroff command, 47
PPD (PostScript Printer Definition) File Manager
  adding new file to system, 375–377
  locating information about available files, 375
  overview of, 372
ppdmgr command, 376–377
ppriv utility, 338
prctl command, 339
preap command, 145, 151–152
Preboot eXecution Environment (PXE), 257–258, 259–261
preconfiguration, 287
Predictive Self-Healing
  comprised of Fault Manager. *See* Fault Management
  comprised of SMF. *See* SMF (Service Management Facility)
  defined, 179
primary group users, 310
print administration, 369–394
  network protocols, 372–373
  planning for printer setup, 373–378
  print servers and print clients, 371
  printer categories, 370
  printer connections, 370
  setting up printer on print client with Solaris Print Manager, 385
  setting up printer with Solaris Print Manager, 379–384
  Solaris print architecture, 369–370
  troubleshooting printing problems, 391–394
  using LP print commands, 385–391
  using LP print service, 371–372
  using PostScript Printer Definition File Manager, 372
  using Solaris Print Manager, 371
Print Application Programming Interface (PAPI), 370
print clients
  defined, 371
  network protocols for, 372–373
  planning for printer setup, 373–374
  setting up printer with Solaris Print Manager on, 385
  troubleshooting printer network connections, 394

print queues, 370, 371
print scheduler, troubleshooting, 392–393
print servers
  defined, 371
  network protocols for, 372–373
  planning for printer setup, 373–374
  troubleshooting printer network connections, 394
printer definitions, Solaris Print Manager, 379–380
printers database, 377–378
printing disk labels, 229–232
privileged users, 310, 328
privileges
  altering in non-global zone, 338
  using DTrace in non-global zone, 339
proactive patch management
  identifying and accessing patches for, 73–75
  during installation of new system, 71–73
  overview of, 62
  Solaris Live Upgrade for, 62–64
  on systems with non-global zones installed, 66–68
  when installing new system, 71–73
  when to use patchadd for, 64–66
problem prevention, with proactive patch management, 62
proc structure, processes, 144–145
procedure scripts, package installation, 48
process contracts, with kernel, 42–43
Process Control Block (PCB), 145
Process Manager, 164–170
processes, 141–178
  context of, 143–145
  contract service, 43
  controlling, 158–164
  overview of, 141–142
  state of, 143
  using Process Manager, 164–170
processes, monitoring, 145–158
  commands for, 145–146
  process arguments, pargs, 152–153
  process file table, pfiles, 153–154
  process libraries, pldd, 154
  process stack, pstack, 155
  process statistics summary, prstat, 149–151
  process status, ps, 146–149
  process tree, ptree, 154
  process working directory, pwdx, 152
  reaping zombie process, preap, 151–152
  resuming suspended process, prun, 152
  searching for process, pgrep, 149
  service, 42–43
  temporarily stopping process, pstop, 152
  tracing process, truss, 156–158
  wait for process completion, pwait, 152
processes, scheduling, 170–178
  at command, 175–178
  controlling access to crontab, 174–175
  cron utility, 171–174
  overview of, 170–171
  removing, 176–177
prodreg viewer, 49, 50
profile file
  creating, 18
  creating profile server, 15
  overview of, 17
profile server, creating
  adding or deleting packages, 18–19
  defined, 14
  profile file, 15
  profile file, creating, 18
  profile file syntax, 17
  rules file, 15
  rules file, creating, 16–17
  rules file syntax, 15
profile server, setting up LDAP client, 356
profiles
  configuring Native LDAP client, 367–368
  user, 312
protocols database, 347
prstat command, 145, 149–151
prtconf command, 236, 238–240
prtvtoc command
  displaying disk partition table, 208–209
  printing disk label, 229–230
  viewing slice information for disk, 205
prun command, 146, 152
ps command, 145, 146–149
pseudo file systems
  defined, 136
  types supported by Solaris OS, 100
  using loopback file system, 139

using swap space, 136–138
using TMPFS file system, 138
pstack command, 146, 155
pstop command, 146, 152
ptree command
ancestry selection in Process Manager vs., 171
defined, 146
displaying process tree, 154
pwait command, 146, 152
pwdx command, 146, 152
PXE (Preboot Execution Environment), 257–258, 259–261

Q

quota -v command, UFS, 110
quotacheck command, UFS, 110
quotas, UFS, 108–109
quotas, ZFS, 120

R

R-patches (restricted patches), 91
RAID-1 (mirroring), 94, 121
RAID-1+0, or RAID 10 (mirroring), 121
RAID-5 (striping with parity), 121
RAID-Z, 122
RAID-Z pool, 122
raw device interface, 202
RBAC (role-based access control)
configuring client hosts in a domain, 282
configuring router for a domain, 280–282
configuring DHCP server, 285
defined, 319
network configuration, 277
reactive patch management, 68–71
read command, 222
README file
installing interactive patches, 92
installing nonstandard patches, 91
overview of, 93
patch dependencies, 96
rebooting after applying or removing patch, 94–95
when to patch in single-user mode, 93
on withdrawn patches, 92
Ready or Runnable state of process, 143
Ready to Install screen, SPARC, 8–9
reboot command, 36–37, 47
rebooting
after applying or removing patch, 94–95
after SPARC Solaris installation, 8
Recommended Patch Cluster
accessing with support contract, 73
installing, 64, 72–73
JumpStart profile for applying, 72
reference platforms, swapping disks, 263–264
registration, new driver, 255
Registry, FMA Events, 194–196
reject command, print requests, 387
rejuvenated patches, 90
releases, Solaris Live Upgrade planning, 77
remote file systems, mounting, 101
remote package servers, 53, 55
remote printer, 370
removable media
mounting, 101
overview of, 133–135
using HSFS, 136
using PCFS, 135
removing packages
pkgrm command, 50–51, 56–57
spooled package, 54, 57
renice command, 159, 170–171
repairing faults, 184, 193
reports, Fault Management, 180
repository, restoring SCF data to, 46
repquota command, UFS, 110
request ID, canceling print request, 388–389
request scripts, packages, 48
resolv.conf file, DNS, 350
resolvers, DNS client, 342, 350–351
resources
Fault Management downloadable, 193–196
NFS server sharing, 127
response agents, Fault Management, 180, 182
restarter, SMF service implementation, 41
restarters, alternative service, 43–44
restart_on value, SMF dependencies, 42
restoring file systems, 107–108
restricted patches (R-patches), 91

RFC-1179 protocol, 372
risk, reactive patch management, 69
rm command, 56
role-based access control. *See* RBAC (role-based access control)
roleadd command, 312
roledel command, 312
rolemod command, 312
roles
  initializing LDAP client, 362
  initializing LDAP client manually, 363
  managing, 318–320
  overview of, 310
root file system, 99
root user. *See* superuser
root_archive command, 261–262
root_device keyword, profile syntax, 17
rotation, log file, 185–187
routers
  configuring in domains, 280–284
  configuring DHCP server, 284–285
  setting up network, 279
routes, adding manually to routing table, 297–298
routing, static vs. dynamic, 295–296
rules file
  creating, 16–17
  creating profile server, 15
  syntax of, 15
  validating, 20
run levels
  boot process and, 37
  corresponding system behaviors, 38
  overview of, 36–37
runat command, 115
running state of process, 143

S

safe privileges, zones, 338
sasl/GSSAPI authentication method, Per-User credentials, 357–359
scanpci command, 240–241, 249
SCF (service configuration facility), SMF
  backing and restoring data, 45–46
  importing service manifests into SCF repository, 45
  overview of, 44
  starting services implemented by daemons, 42–43
scheduling processes. *See* processes, scheduling
schemas, ldapclient init, 365
script patches, 60, 92
secondary (non-system) disks, 208
sectors, disk
  defined, 199–200
  identifying defective, 221–222
  repairing, 222
Secure by Default, 4
security
  free patches for, 73
  proactive patch management for, 62
  Sun xVM Ops Center patch automation tools, 86–87
  using Interim Security Relief patches, 91
  using Secure by Default, 4
Security T-Patches, 90
Server Message Block (SMB), 373
servers, consolidating. *See* zones
service configuration facility. *See* SCF (service configuration facility), SMF
Service Management Facility. *See* SMF (Service Management Facility)
service manager, SMF
  dependencies, 41–42
  enabling/disabling services, 40
  logging information about service events, 41
  overview of, 39–40
  service states, 40–41
  starting services implemented by daemons, 42–43
service manifests, SMF, 45
services database, 347
setfacl command, 112–113
shared-IP zones, 324–325
shell, login, 312
Show Ancestry option, Process Manager, 170–171
showmount command, 128–129
shut down
  application-independent, 46
  application-specific, 46
shutdown command, 47, 85

SIGINT signal, through Process Manager, 169
SIGKILL signals, 43, 47
signal handlers, 160
signals, 159–164
  defined, 159
  list of common, 161–163
  sending to process, 159–160
  signal handlers, 160
  signal masks, 161
  states of, 160–161
  using kill command, 163–164
SIGTERM signal, 47
Simple Network Management Protocol (SNMP), 182
Simultaneous Peripheral Operations On-line (spooler), 369
single-user mode, system patches, 93
slave servers, NIS
  creating list on master of, 355
  defined, 343
  NIS setup and configuration, 351
  updating map, 354–355
sleep state of process, 143
slices. *See* disk slices
SMB (Server Message Block), 373
SMC (Solaris Management Console), 312, 314–315
SMF (Service Management Facility)
  backup and restore of SCF data, 45–46
  checking status of services, 338
  dependency, 41–42
  enabled or disabled, 40
  enabling network listening services, 378
  Fault Manager daemon under, 180
  health and troubleshooting, 44–45
  interacting with service implementations, 42–44
  logfile, 40–41
  login, 35
  overview of, 39–40
  restarter and contract_id, 41
  service configuration facility, 44
  service manager, 39–40
  service manifests, 45
  state, next_state, and state_time, 40–41
smgroup command, 312, 316–317
SMI disk labels, 203, 204
smpatch command, 87
smrole command, 312
smrole list command, 319–320
smuser command, 312, 316–317
snapshots
  UFS, 115–116
  ZFS, 122–123
SNMP (Simple Network Management Protocol), 182
snmp-trapgen module, 182
snoop command, 306
Software Groups, 4–5
software interrupts. *See* signals
software management
  packages. *See* packages, software
  patches. *See* patches
Solaris 10 Live Upgrade Patch Bundle, 67
Solaris 10 OS installation
  basics of, 2–5
  initial install checklist, 28–31
  JumpStart method of. *See* JumpStart installation
  methods to meet your needs, 1–2
  performing JumpStart custom installation, 22–25
  reference information, 27–31
  Solaris Live Upgrade, 26–27
  SPARC system installation, 6–9
  upgrading, 25–26
  x86 system installation, 9–13
Solaris Containers. *See* zones
Solaris for x86 Device Support database, 249, 250
Solaris Installation Program Welcome screen, SPARC, 6–7
Solaris Installation screen, SPARC, 7–8
Solaris Live Upgrade, 75–86
  applying patch for Solaris 10 8/07, 79–86
  applying proactive patches, 63
  applying reactive patches, 70
  applying security patches, 71
  defined, 2
  example of installing patches, 75–76
  overview of, 26
  patching on systems with non-global zones, 66–67, 70
  planning and limitations, 75–76

Solaris Live Upgrade *(Continued)*
rebooting after applying or removing patch, 94
as recommended patching tool, 62–63
upgrading ZFS root pool, 24
when to patch in single-user mode, 93
Solaris Management Console (SMC), 312, 314–315
Solaris Print Manager
overview of, 371
setting up printer on print client, 385
setting up printers, 379–384
Solaris Text Installer Welcome screen, SPARC, 7
Solaris zones. *See* zones
source files, naming services, 290–291
space files, packages, 48
SPARC systems
accessing patches, 73–74
applying patch for Solaris 10 8/07, 85
custom JumpStart installation on, 22–23
installing Solaris on, 6–9
sparse packages, 59
sparse-root zones, 323
spool directory
copying frequently installed packages to, 54–55
installing software package from, 56
removing spooled software package, 57
spooler (Simultaneous Peripheral Operations On-line), 369
spooling space, printer setup, 374
stack trace, 155
standard installation, 2
start method, 42–43
`state`, SMF services, 40–41
states, process, 143, 148–149
states, signal, 160–161
`state_time`, SMF services, 40–41
static routing, 295–298
statistics, tracking with Fault Manager, 191–192
status, database actions and, 349
stop methods, daemons, 43
storage pools, ZFS
checking, 126–127
defined, 117
expanding, 125–126
overview of, 118–120
using snapshots, 123
`strings` command, 247
striping, ZFS, 121
`stty` command, 22
subnets, installation with, 3
subnetworks, 278
Sun Alerts
proactive patching with, 63
Recommended Patch Cluster containing, 72–73
registering for, 71
for withdrawn patches, 92
Sun Device Detection Tool, 237–238, 249
Sun Explorer logs, 69
Sun Solve website, 67
Sun xVM Ops Center, 86–87
`SUNW` prefix, 49
`SUNW_IMCOMPAT` field, 96, 97–98
`SUNW_OBSOLETES` field, 96, 97
`SUNW_PKG_ALLZONES` package parameter, 326, 333
`SUNW_PKG_HOLLOW` package parameter, 326
`SUNW_PKG_THISZONE` package parameter, 326
`SUNW_REQUIRES` field, 96
superblocks, UFS, 106, 112
superuser
changing `root` from user to role, 318
configuring network as, 277
defined, 310
initializing LDAP client as, 362
initializing LDAP client manually, 363
supplementary group users, 310
support contract, Solaris patches, 73
surface analysis, identifying defective disk sector, 221–222
suspect list, Fault Management, 180
`svcadm` command, 378
`svcadm disable` command, 135
`svccfg archive` command, 46
`svccfg delete` command, 47
`svccfg extract` command, 46–47
`svccfg import` command, 46–47
`svcs` command, 40, 146
`svcs -l` command, 43–44
`svcs -p` command, 43
`svcs -x` command, 44

SVR4 packages, 47. *See also* packages, software
SVR4 packages, installing, 252
swap slices, 136
swap space, 136–138, 374
swapped caps, zones, 339
swapping disks, 263–265
switch file, naming service. *See* `/etc/nsswitch.conf` file
`sysidcfg` file
  defined, 13
  preconfiguration and, 287
  setting basic network configuration, 14
`syslog` message, 358–359
system administration tasks, x86 Solaris installation, 11–12
system disk, 208
systems and system names, networks, 277–279

T

T-patches (temporary patches), 89–90
`tar` command, UFS snapshot, 116
TCP/IP networking stack, 275–277
TCP (Transmission Control Protocol), 372
temporary file system. *See* TMPFS (temporary file system)
temporary patches (T-patches), 89–90
`terminfo` database, 374–375
testing
  network configuration, 301–302
  new driver, 256
  patches in T-patch state, 89–90
  profile server, 19
text editor, 18
text installer
  Solaris OS single-system install, 2–3
  SPARC Solaris installation, 6–7
  x86 Solaris installation, 11
third-party applications
  disk slice limitations, 206
  Solaris Live Upgrade patch limitations, 78
time stamps, problems in driver code, 269
`tip` window, custom JumpStart installs, 22–24
TMPFS (temporary file system)
  defined, 100
  using, 138
  using extended file attributes, 115
`traceroute` command, 306–307
tracing process, `truss`, 156–157
tracks, disk, 199–200
transaction logs, UFS, 114
transient service model, 43
Transmission Control Protocol (TCP), 372
transparent compression, ZFS, 124
troubleshooting
  boot process, 37–39
  LDAP client configuration, 364
  naming services, 368
  printing, 391–394
  Service Management Facility, 44–45
  zones, 337–339
`truss` command
  defined, 146
  reactive patching using, 69
  tracing process, 156–158
`ttymon` program, 34–35

U

u area, context of process, 144
UFS (UNIX file system), 105–116
  backing up and restoring, 107–108
  checking integrity, 110–112
  creating, 106–107
  creating snapshots, 115–116
  defined, 100
  overview of, 105–106
  repairing inconsistencies using `fsck`, 232–233
  upgrading Solaris, 24
  using access control lists, 112–113
  using extended file attributes, 115
  using logging, 113–114
  using multiterabyte, 115
  using quotas to manage disk space, 108–109
  ZFS for boot file system vs., 5
UID (user ID), 312
`umount` command
  identifying defective disk sector, 221

umount command *(Continued)*
unmounting file systems, 101
unmounting NFS, 121
umountall command, 101
underprivileged users, 310
Uniform Resource Identifier (URI), IPP and, 373
uninitializing LDAP client, 364
uninstalling zones, 333
UNIX file system. *See* UFS (UNIX file system)
unmounting file systems
with automounting, 130–133
identifying defective disk sector, 221
NFS, 129
overview of, 102–103
UFS transaction log flushed after, 114
Update Manager GUI, patches, 87
Update releases, 92
upgrade
correcting modifications not preserved during, 8
custom JumpStart on SPARC, 22–23
methods, 2
overview of, 25–26
using Solaris Live Upgrade, 26–27
upgrade_cleanup file, 8
URI (Uniform Resource Identifier), IPP and, 373
USB memory stick, accessing information, 134–135
user ID (UID), 312
user management, 309–320
file system object permissions, 310–311
files, 313–314
groups, 310
managing roles, 318–320
managing users and groups, 314–317
overview of, 309
roles, 310
tools, 313
user account components, 312–313
users, 309–310
user-mode programs, boot process, 34–35, 39
user name, 312
useradd command
adding NIS users to NIS domain, 294
adding user to local files, 315–316
defined, 312
userdel command, 312
usermod command
assigning role to local user, 320–321
changing root from user to role, 318–319
defined, 312
users
adding to local files, 315–316
adding to NIS domain, 316–317
assigning role to local, 320–321
file system object permissions, 310–311
managing, 314–317
overview of, 309–310
profiles, 312
UFS file system ACLs specifying, 112–113
/usr/bin/disable command, 387
/usr/bin/sh command, 312
/usr/lib/lp/ppd/system directory, 375
/usr/sbin/ppdmgr command. *See* PPD (PostScript Printer Definition) File Manager
/usr/sbin/ypinit -m command, 355
/usr/share/lib directory, 374–375

V

validation, rules file, 20
variables, 19
vendor ID, PCI devices, 236–237
Veritas Storage Foundation root disk, 64, 94
Veritas Volume Manager (VxVM) volumes, 94
version information, driver, 246–247
viewing, zone configuration, 331
virtual consoles, zones, 332–333
vmstat command, 269
volcheck command, 101, 134
vold daemon, 133–134
VTOC (Volume Table of Contents) disk labels
creating disk labels, 224–225
displaying slice information, 223–224
EFI vs., 204–205
overview of, 203
partition table for, 209
printing using fmthard, 230–232
printing using prtvtoc, 229–230
slice configuration for, 205–207
VxVM (Veritas Volume Manager) volumes, 94

W

whole-root zones, 323–324
withdrawn patches, 92
workstations, and DNS namespace, 342–343

X

x86 systems
  accessing patches, 73–74
  applying patch for Solaris 10 8/07, 85–86
  custom JumpStart installation on, 23–25
  `fdisk` keyword, 19
  `fdisk` partitions in, 206
  identifying disks, 214–215
  installing Solaris on, 9–13

Y

`ypbind` command, 351–352
`ypcat` command, 353
`ypinit` command, 355
`ypinit` method, 351
`ypmatch` command, 353–354
`ypwhich` command, 353

Z

`zfs create` command, 118–119, 122
`zfs destroy -r` command, 120
ZFS file system, 117–127
  backing up, 120
  checking pool, 126–127
  cloning zones, 334
  defined, 100
  expanding pool, 125–126
  measuring performance, 124–125
  mounting local, 101
  overview of, 117–118
  replacing disk, 127
  using copy-on-write and snapshots, 122–123
  using extended file attributes, 115
  using file compression, 124
  using mirroring and striping, 121
  using pools, 118–120
  using RAID-Z, 122
`zfs get all` command, 119–120
`zfs get compressratio` command, 124
`zfs list` command, 124
`zfs mount` command, 101
`zfs receive` command, 120
ZFS root pools
  booting from, 26–27
  `grub` reading Solaris, 36
  selecting UFS vs., 5
  upgrading Solaris system, 24
`zfs scrub` command, 126–127
`zfs send` command, 120
`zfs set` command, 124
`zfs snapshot` command, 120, 122–123
`zlogin` command, 332–333
zombie process, 143, 151–152
`zoneadm` command, 331–332, 334–335
`zoneadm halt` command, 333
`zoneadm uninstall` command, 333
`zonecfg` command
  adding dedicated device to non-global zone, 337
  configuring zone network interfaces, 324
  creating zone, 327–331, 333
  defined, 325
  migrating zone, 334–335
  modifying CPU, swap and locked memory caps in zones, 339
  using DTrace in non-global zone, 339
  viewing zone configuration, 331
`zonename` property, non-global zones, 327–328
`zonepath` command, 327–328, 333, 334–335
zones, 321–339
  branded, 324
  cloning, 333–334
  configuring, 327–331
  customizing and troubleshooting, 337–339
  deleting, 336
  devices in, 325
  halting, 333
  installation and booting, 331–332
  listing on system, 336
  login using `zlogin` command, 332–333
  migrating to new system, 334–336
  moving, 333
  network interfaces in, 324–325
  overview of, 321–322

zones *(Continued)*
packages and patches in environment, 325–326
understanding, 323–324
uninstalling, 333
viewing configuration, 331
`zpool add` command, 125
`zpool attach` command, 121
`zpool` command, 118
`zpool iostat` command, 124–125
`zpool list` command, 118
`zpool replace` command, 127
`zpool status` command
checking pool, 126
expanding pool, 125
measuring performance, 124–125
replacing disk, 127
using mirroring and striping, 121