BrainBuzz

Cramsession

Last updated October, 2000.
Click here for updates.

Click here to see additional documents related to this study guide.

Cramsession™ for Sun Solaris 7 Certified Systems Administration I

Abstract:

This Cramsession will help you to prepare for Sun Exam 310-009, Sun Solaris 7 Certified Systems Administration I. Exam topics include: Stand-alone Installation, File System Management, Process Control, User Administration, Device Management, Systems Concepts, Boot PROM, Software Package Administration, Patch Maintenance, Security, Adding Users, Initialization Files, Process Control, Disks, Partitions and Formating, Mounting, Backup and Recovery and Print Service.

Contents

System Concepts

Match the three parts of an operating system (kernel, shell and file system) to their definitions

Kernel – Acts as an intermediary between applications running on a computer and the hardware inside the computer. It controls physical and virtual memory, schedules processes, and starts and stops daemons. All commands interact with the kernel.

Shell – The shell interprets and translates commands entered by the user into actions performed by the system. There are three by default in Solaris: Bourne Shell, Korn Shell, and C Shell.

File System – Data on a Solaris system is stored in a hierarchical fashion on the file system. Organizing data in this way makes it easy to locate and group related operating system control files and user information.

Identify the three most common shells in the Solaris environment

Feature	**sh**	**csh**	**ksh**
Aliases	No	Yes	Yes
Command line editing	No	Yes	Yes
History capability/editing	No	Yes/Yes	Yes/No
History execution	No	!*n*	r *n*
Prompt*	$	*system name*%	$
Repeat last command	No	!!	
Initialization file - login	.profile	.login	.profile
Initialization file - shell startup	No	.cshrc	user defined

Distinguish between multitasking and multiuser

Multitasking - A CPU executes multiple processes in very quick succession, allowing unrelated applications to share the processor, and giving the user the impression that programs are being run simultaneously.

Multiuser – Users share a computer system by running their own copies of program processes on the CPU.

Describe the client-server relationship

A client-server (C/S) setup is a network environment where server hosts or processes provide services to client hosts or processes. A client-server environment is known as distributed processing. Examples of shared services include: file sharing, electronic mail, printing, and NIS.

Define the following basic system terms: host, host name, network, IP Address, client, server

Host – A networked computer system
Host Name – A logical alphabetical name given to a computer system
Network – A collection of hardware that facilitates communication and shared services
IP address – A logical, unique numerical name used to identify a computer system on a network
Client – A computer on a network that uses shared services
Server – A computer on a network that provides some type of service

The Boot PROM

Use OpenBoot PROM commands to identify basic system configuration information

help – shows commands and their usage

banner – this command lists information relating to the amount of system memory, system model, PROM version, memory and hostid

printenv – lists the parameters of NVRAM and their current values

Use OpenBoot PROM commands to alter the system boot device

The boot command is used to change the system boot device. There are several included by default: cdrom, disk, tape, floppy, net. Example:

```
ok boot net
```

will start the system booting from a kernel it obtains from a boot server.

The default device is `disk`. This can be changed by setting the `boot-device` parameter:

```
# setenv boot-device <device>
```

Use OpenBoot PROM commands to perform basic hardware testing

Command	**Purpose**
`probe-scsi`	Test the built-in SCSI bus for connected devices
`probe-scsi-all`	Test all SCSI buses
`test-all`	Test a group of installed devices
`test floppy`	Test diskette drive
`test /memory`	Test memory
`test net`	Test on-board Ethernet controller
`watch-clock`	Test system clock
`watch-net`	Monitor network connection

Boot the system from more than one device

`devalias` – run at the OK prompt to show available boot devices

`boot` – run this command followed by a name from `devalias` to start a system initialization

`auto-boot` – this command, off by default, enables or disables the system from automatically booting from the default device after a system power-on or restart

Create a custom device alias using nvalias

If the *use-nvramrc* parameter is set to **true**, then the script is executed during start-up. The script editor *nvedit* can be used to copy the contents of the script into a temporary buffer where it can be edited. After editing, the *nvstore* command can be used to copy the contents of the temporary buffer to *nvramrc*. The *nvquit* command is used to discard the contents of the temporary buffer.

The alias defined by the *nvalias* command remains in the script until either the *nvunalias* or *set-defaults* command is executed. The *set-defaults* command can be undone by the *nvrecover* (if the script has not been editied).

```
ok nvalias <custom name> \
/iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/esp@5,8800000/sd@3,0

ok setenv boot-device <custom name>

ok boot
```

Any aliases defined by the *devalias* command are lost during a reboot or system reset. Aliases defined by the *nvalias* command are not lost.

Remove a custom device alias using nvunalias

```
ok nvunalias <custom name>

ok setenv boot-device disk

ok reset
```

Use the Solaris™ eeprom command to modify EEPROM parameters

As the superuser, and while the operating system is running, the `/usr/sbin/eeprom` command can be used to query and change NVRAM values. Changes are permanent.

Use the boot command options to observe system boot problems

`boot –a` – starts a boot sequence in interactive mode, which allows the user to specify root and swap volumes, as well as system configuration files

`boot –r` – probes all devices attached to the system and updates the `/devices` and `/dev` file trees. Useful after adding new devices to the system

`boot –s` – starts the operating system into single user mode. Useful for bringing the system down for administration

`boot –v` – displays verbose and detailed startup messages

Use keyboard commands to abort a hung system

Stop-a – will interrupt the running operating system and return to the OpenBoot OK prompt

Stop-n – holding down this sequence while the system is booting will reset the values of the NVRAM to the factory defaults

Stop-d – this key combination will run the diagnostic mode (equivalent to `diag-switch?=true`) when the system boots up

Installing the Solaris™ 7 software on a stand-alone system

Define software configurations, clusters, and packages

1. Software Configuration – Three types:
    - Core – The base install, containing drivers
    - End User – Core + OpenWindows and CDE
    - Developer – End User + compiler tools and man pages
    - Entire Distribution – All of Solaris 7, plus OEM packages
2. Clusters – collections of similar software, usually named SUNW<packageabbrev>
3. Packages – a group of files and directories that make up a particular application

Identify the hardware requirements for installing the Solaris™ 7 software on a standalone workstation

SPARC or Intel system

1.05 Gigabytes of free space

64 Megabytes of System Memory

CDROM drive or network access to a Jumpstart™ server

Prepare an existing system for a standalone installation

Log in as root

Have all users save files and log off the system

Back up necessary user files or configuration information

Shutdown the system to the OK prompt

Insert the Solaris 7 CDROM, and type boot cdrom

Install the Solaris™ 7 software on a standalone workstation using SunInstall™

Suninstall is a graphical Xwindows tool that prompts the user for system configuration information using dialog boxes, prompts, and radio buttons.

Software Package Administration

Display software package information

The pkginfo command is used to check the installed packages on the system.

`pkginfo [ -d [device | pathname] ] [-l] package_name`

Example: `pkginfo –d /cdrom/cdrom0/s0/Solaris_2.7/Product`

Add a software package from a CDROM drive

The pkgadd command is used to install a package from an installation source

`pkgadd [ -d [device | pathname] ] package_name`

Example: `pkgadd –d /cdrom/cdrom0/s0/Solaris_2.7/Product SUNWaudio`

Remove a software package

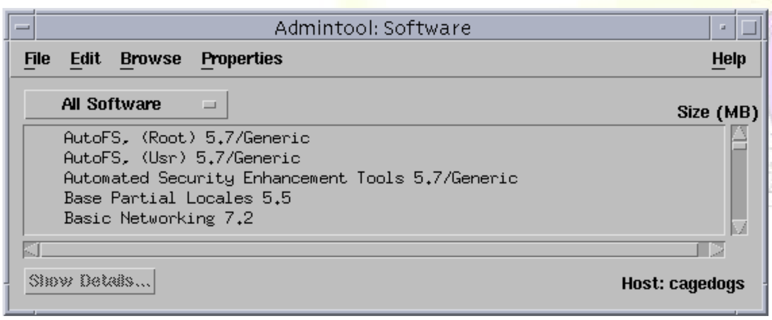
Use the pkgrm command to remove an installed package

`pkgrm package_name`

Example: `# pkgrm SUNWaudio`

Add and remove software packages using the Admintool™ software program

Run `admintool` as superuser. Choose software from the browse menu.



Select package names in the display window, use the edit menu to add or delete a package.

Add a software package from a spooled directory

Spooled software describes software packages that are copied from a CDROM to a default location on disk for later installation. The default in Solaris™ is `/var/spool/pkg`. An administrator adds a package to the spool directory using the –s spool switch. Example:

`# pkgadd –d /cdrom/cdrom0/s0/Solaris_2.7/Product –s spool SUNWaudio`

To add software from the default spool directory, run `pkgadd <package_name>`

Maintaining Patches

Obtain current patch information and patches

A patch refers to a collection of files used to update or replace existing installed system software. They may be obtained from Sun using sunsolve.sun.com via the WWW or FTP, or from other trusted third parties.

Determine that you need the latest patch by reading patch reports. Download the appropriate software to your system.

Verify current patches installed on your system

On a Solaris 7 system, use the patchadd –p command to view installed patches. A legacy command, showrev –p, will also display the same information.

Install patches

Copy the appropriate patch software to /tmp. Execute the command patchadd <patchname>. Check the log file in /var/sadm/patch/<patch_name>/log for details of the installation

Back out patches

Use the command patchrm <patch_name> to remove an installed patch from the system. All files modified by the patch are restored unless:

- The patch was installed using patchadd –d
- The patch was rendered obsolete by a later patch
- The patch is required by another patch

The Boot process

Describe the functionality available at each of the eight system run levels

The Solaris™ operating system uses run levels to describe certain states of the overall machine. An administrator must be aware of the functionality provided at each level to ensure that the system runs smoothly.

Run Level	State	Functionality
0	Power-down	Safe to turn off power to the system.
1	Administrative Single-user	All available file systems with user logins allowed. The terminal from which you issue this command becomes the Console.
2	Multiuser	For normal operations. Multiple users can access the system and the entire file system. All daemons run except for NFS server and syslog.
3	Multiuser w/ NFS	For normal operations with NFS resource-sharing available.
4	Alternative multiuser	This level is currently unavailable.
5	Power-down	Shutdown the system and automatically turn off system power (if possible).
6	Reboot	Shutdown the system to run level 0, and then reboot to multiuser state (or whatever level is the default in the *inittab* file).
S, s	Single-user	Single user mode with all file systems mounted and accessible.

List the phases of the boot process

1) Self-test and POST diagnostics are run
2) System identification via the banner is displayed
3) The disk label at sector 0 is read
4) The primary boot program, `bootblk`, is loaded to read the UFS file system. It was placed there using the `installboot` command during installation
5) `bootblk` loads the boot program into memory from: /platform/‘uname –m`/ufsboot

Explain the main roles of the /sbin/init program

After the kernel is loaded into memory, it begins an initial process called `/sbin/init` which is charged with starting all of the daemon processes. It reads a file `/etc/inittab` to learn about what it needs to do for each process.

The `/sbin/init` will also reread the `/etc/inittab` file when the system changes runlevels. Special scripts known as rc scripts are called at each run-level.

Add startup files for additional system services

Choose a run level for the service you are adding. Create a startup (‘S’) and shutdown (‘K’) script in the `/etc/init.d` directory. Change to the `/etc/rc#.d` directory (# = runlevel) and link to the corresponding `/etc/init.d` file for both the startup and shutdown scripts. Restart the system into the new desired runlevel.

Changing system states

List at least two reasons for halting a system

Solaris, unlike other operating systems, is designed to be reboot-free. However, certain situations require the system to be brought down. The most obvious is when adding and removing hardware. Another might be to install a new release of Solaris. Solaris must be brought down when backing up or restoring certain kinds of data.

List the five commands used to change system run levels from the command line

```
init
shutdown –i
halt
/usr/sbin/poweroff
reboot
```

Change run levels using the init and shutdown commands

To change run levels using the init command, execute it as root followed by the desired runlevel. Example:

`# init 6` (will reboot the system)

Using the shutdown command to change run-levels occurs when the `–i` option is specified: `shutdown –i <init-state>`

System Security

Use the id command to determine your UID (user identifier) and GID (group identifier)

`$ id` (will display all relevant user and group information)
`uid=0(root) gid=1(other)`

`$ id –a` will show user information, and all groups to which the user belongs

Describe the superuser account and its importance to system administration

The user with UID 0 is the root or super user. This account is granted read and write access to all files on the disk, and can kill all processes run by the CPU. These abilities make it the perfect account to perform system tasks like power-up and power-down, backup and restore, and adding users and file systems. Because of the few limitations imposed, the superuser account should be used infrequently and be closely guarded.

Describe the purpose of the sysadmin group

The sysadmin group has rights to modify system control databases, by using the `admintool` and Solstice™ Adminsuite™ products. A regular user account can be added to this group to perform system administration functions without being given root access.

Change user ownership of files and directories

The `chown` tool is used to change ownership of files from user to user. The creator owns new files by default. The superuser uses the `chown` command to reassign the files ownership rights.

```
# chown username filename
```

Change the group ownership of files and directories

The `chgrp` command is responsible for this system function. It changes the group rights for files and directories (perhaps when transferring existing data to new user accounts).

```
# chgrp groupname filename
```

A quick shortcut for changing file and group ownership is to use the `chown` command, but specify the username:groupname delimited by a ":".

```
# chown <username>:<groupname> <filename>
```

Describe how the who and last commands relate to system security

The who command shows the usernames that are currently logged into the system. it displays name, login device, login time, and remote system name (if applicable).

```
$ who
root       console      Sep 17 23:24    (:0)
root       pts/4        Sep 17 23:25    (:0.0)
root       pts/3        Sep 17 23:25    (:0.0)
root       term/a       Sep 26 08:19
```

The last command shows the most recent login and logout activity. By default, it displays name, login device, system logged in from, date and time logged in, time logged out, and total login time in hours:minutes.

```
$ last
root      term/a                       Tue Sep 26 08:19  still logged in
root      console      :0              Mon Sep 11 20:48 - 05:15 (1+08:26)
mattk     pts/5        localhost       Mon Sep 11 13:39 - 20:47  (07:07)
josephh   pts/5        localhost       Mon Sep 11 13:36 - 13:36  (00:00)
reboot    system boot                  Thu Sep 14 02:08
```

Describe the format of the /etc/passwd, /etc/shadow, and /etc/group files and explain their importance to system security

The /etc/passwd file contains a record of basic user account information for each user on the system. An entry for the user must be in this file, or login attempts will be invalid

```
loginID:x:UID:GID:comment:home_directory:login shell
```

The /etc/shadow file contains the actual encrypted password entry for the user account. The 'x' in the second (password) field of the /etc/password file is a placeholder that references this file. The file is not edited by the superuser, rather, its entries are created using the passwd utility or the admintool. The fields after the password describe the password aging scheme for that particular account.

```
loginID:password:lastchange:min:max:warn:inactive:expire:
```

The /etc/group file maintains a record of all of the groups defined on a system. The file establishes a relationship between the numerical GID and a name assigned by the administrator. The file may be manually edited, but the preferred method is using the admintool or groupadd.

```
groupname:password:GID:userlist
```

Modify several system default files that enable the system administrator to control and monitor superuser access to the system

Administrative rights to the system files may be delegated by adding a user account to the /etc/groups file for the sysadmin (GID=14) group.

A normal user may assume root identity during a session by using the su command. The user must specify the root password, however. The su activity is logged in the /var/adm/sulog file.

A system adminstrator may also monitor who is logged in as {superuser} by using the who command. For a login history, the last command may be run.

Restrict access to the root account

root access may be restricted to the console only by modifying the /etc/default/login file, and removing the # comment from the CONSOLE variable. A system administrator may also take basic precautions by changing the root password after a set number of months.

Describe how to monitor logins

Active logins are monitored using the who command. For remote access to login information, the finger command may be used.

The last command records those users that were logged into the system and for how long. It also logs the times the system was restarted.

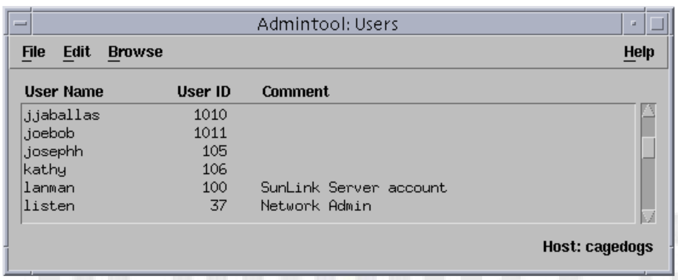
The administrator may examine the `/var/adm/sulog` file to see who was using the su command to obtain root access.

Adding users

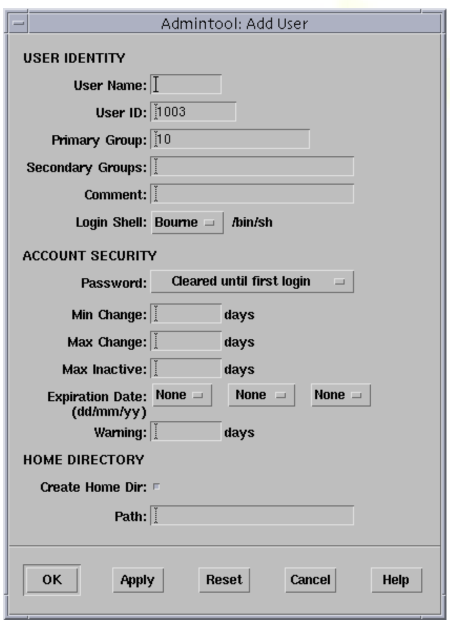
Use Admintool™ to create a new group and a new user account

`admintool`, a graphical administration tool, is provided with Solaris™ 7 to ease the management of necessary tasks on the system. `admintool` manages users, groups, hosts, printers, serial ports and software packages.

Start `admintool`, and select Users from the Browse menu:



In the Edit Menu, select Add:



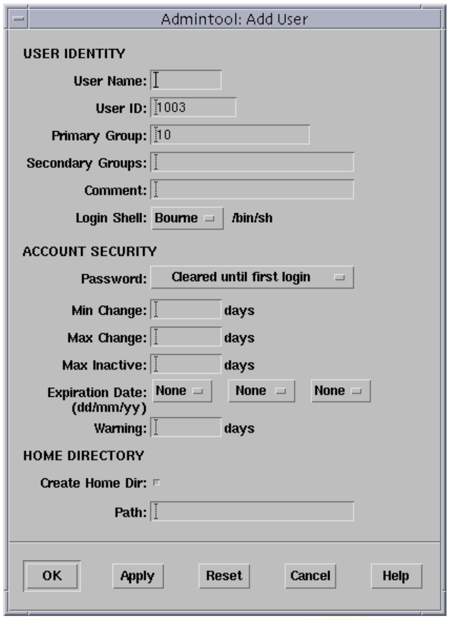
Fill in the blanks and choose OK to add the user.

Change your password

Use the `passwd` tool at the command line. It will prompt you twice.

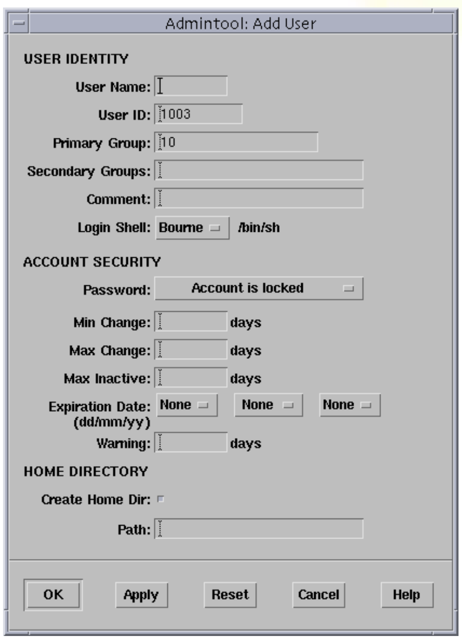
Setup password aging using Admintool™

To set password aging, enter values into the fields found in the Account Security area of the User Add dialog:



Lock a user account using Admintool

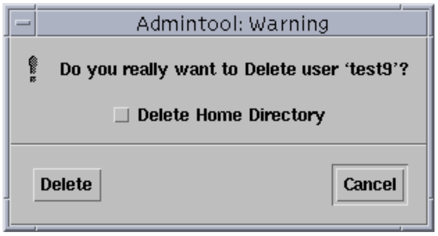
In the account security password dropdown menu, select "Account is locked":



Verify that `*LK*` appears in the password field for the user in `/etc/shadow`.

Delete an account using Admintool™

Select the name of the user, and then choose Edit | Delete. To remove the user's configured home directory, click the radio button in the prompt that appears.



An administrator may wish to save the user's home directory and files for a period of time after the account name is removed.

Administration of Initialization files

Define a variable in the .profile file

Shell variables are values that set up certain aspects of the user's shell environment, such as the `DISPLAY` or `PATH`. In order to be referenced, shell variables must first be defined. Open the .profile file with an editor like vi:

```
# vi .profile
```

create a single line of the format:

```
VARIABLE=value
```

for example, `PATH=/usr/local/bin:/usr/ccs/bin`

Multiple VARIABLES may be defined in `.profile`.

Maintain the /etc/profile file

`/etc/profile` is read by the system first, when the bourne shell is in effect. It performs functions like exporting environment variables, setting the `TERM` type, exporting the `PATH`, displaying `/etc/motd`, and setting default permissions. Values set by this file may be overwritten by a user's .profile, because it is read first.

Customize the templates in the /etc/skel directory

The files in `/etc/skel` are `local.profile`, `local.login` and `local.cshrc` (or `local.kshrc` in the case of the Korn shell or Bourne shell). When a new account is created, these files provide the defaults for the .<filename> initialization files in the home directory. Tools like admintool copy these files to a home directory and strip off the "local" prefix.

Customize the initialization files

A sample local.profile file may look like:

```
PATH=/usr/sbin:/sbin:/usr/dt/bin
EDITOR=vi
LPDEST=home_printer
ENV=$HOME/.kshrc
```

Modify the DTSOURCEPROFILE variable so that shell initialization files are read

Terminal sessions that are started within the CDE can be configured to read the current .login or .profile files by configuring the DTSOURCEPROFILE variable:

DTSOURCEPROFILE=true

This line is set in the last line of the .dfprofile file.

Advanced File Permissions

Display and change the default permissions (umask)

`umask` is a variable that determines the default permissions for newly created files. To view the umask, simply type `umask` at the shell prompt. Its default is 022.

umask is represented by three digits. A simple explanation of file permissions: 4 represents read, 2 represents write, 1 represents execute. A combination of these values yields the permission for the owner, group, or 'world' respectively. Example: a file with permissions read/write owner, read/write group, and read-only world would be represented as octal 664 and would look like `-rw-rw-r--` in an `ls` listing of the mode.

To obtain the default file permissions, umask is applied to 777 for directories and 666 for files. Therefore, a new file created with the default umask would have a permission value of 644 (666 – 022).

To change the default value of the umask, add an entry to the .profile similar to the following:

```
# umask <new octal value>
```

Set access control lists on files

Access control lists provide a mechanism for finer control over the basic UNIX file permissions. A file has an acl set if it's output in the `ls` has a '+' at the end of the permissions mode field.

The `setfacl` command is used to set a file ACL.

```
setfacl options acl_entry filename1 [ filename2 filename3 … ]
```

For example, to give read access to another user on a file you own:

```
setfacl –m user:reader1:6 newtext.txt
```

would assign read only access to the user reader1

To verify that ACLs were set for the file, run `getfacl`

To remove an ACL from the file, run:

```
setfacl –d user:reader1:6 newtext.txt
```

Explain how the setuid and setgid permissions relate to system security

`setuid` and `setgid` provide a mechanism for specifying permissions a file must use when being executed, instead of using the defaults which usually come from the process or shell that opened it. If a program runs with `setuid` active, anyone who executes it (as long as they have permission) is treated as if they own the file. `setgid` treats the execution as if the user belonged to the program's assigned group.

`setuid` and `setgid` show up in permissions mode listings as having an 's' in the execute field. I.E. `–r-sr-sr-x` would indicate this file has a `setuid` and `setgid` active.

To enable `setuid` or `setgid`, use the `chmod` command and preface the numeric permissions assignment with a 4 to set the `setuid` or a 2 to set the `setgid`:

```
chmod 4744 setuid_program
chmod 2744 setgid_program
```

For directories, the symbolic notation for setting permissions must be used:

```
# chmod g+s directory_name
```

Identify and set the sticky bit

The sticky bit is set when the permissions mode listing shows a 't' in the execute field for others. It is set by using `chmod` and prefacing the assignment with a 1:

```
# chmod 1777 /var/tmp
```

Describe how the sticky permission protects files and directories

When the sticky bit is set on a directory, it may be publicly write-able, but only the user who creates files in the directory has access to them. This is to safeguard a user's files from being deleted when they are stored in a shared public space like `/var/tmp`.

Process control

Use the ps command to list processes running on a system

Use the `ps` command to display the active system processes and their corresponding process IDs (PIDs). The `–e` option shows every process, and the `–f` option shows a full listing.

Use the kill command to terminate processes running on the system

Use the `ps` command to obtain the PID of the process that needs to be terminated. Run `kill`:

```
# kill <PID>
```

Use the pgrep and pkill commands to locate processes and kill them depending on specified criteria

The `pgrep` and `pkill` commands combine the functionality of `grep`, `egrep`, `awk` and `kill` that were necessary in pre-Solaris™ 7 releases.

`pgrep` will examine the running processes and return those that match certain criteria. It can look for things like UID (`-u`), terminal (`-t`), process name (`-l`), GID (`-G`) and most recent version (`-n`). Example:

```
# pgrep –lf mail
```

returns

```
1924 /usr/dt/bin/dtmail
2412 /usr/lib/sendmail –bd –q15m
```

all the processes that have mail in the name.

Using `pkill` to terminate the processes:

```
# pkill –U root mail
```

will kill all the processes owned by root with mail in the name.

Use the at command to execute a command at a future time

`at` allows tasks to be scheduled to run ONCE at a specified time in the future.

```
at [-m] [-r job] time [date]
```

```
$ at 12:00PM
at>banner “LUNCH!”;
at>^D
```

commands will be executed using `/bin/csh`

Display the job in the at queue using `atq`:

```
$ atq
Rank  Execution Date       Owner  Job        Queue  JobName
1st    Oct 1, 2000   12:00  mk    9342314.a   a        stdin
```

and use `at –r` <jobnumber> to remove the job.

State the function of the cron daemon

`cron` is a system daemon that executes commands to run more than just a single time. It reads entries in the `/var/spool/cron/crontabs` directory. It is used to schedule system maintenance tasks, like removing log files that grow too large.

Describe the format of the crontab file

It is important to remember the layout of the fields in the cron file:

```
30 10 * * 6    /<path to command>
```

From left to right:

the minute
the hour
the day of the month
the month of the year
the day of the week
the command to execute

A * indicates that this value isn't being used ('every'). The above snippet can be read "every Saturday at 10:30 in the morning"

Name the two files used to control crontab access

`/etc/cron.allow` – only users listed in this file may use cron

`/etc/cron.deny` – this file is checked if there is no cron.allow to see if the user is allowed to run cron

If neither file exists, only root may use `cron`

Edit a user's crontab file to schedule nightly backups of the user's home directory

Become the root user (su or by logging in). Run `crontab –e` to start editing the cron session. A sample entry to perform the above request:

```
00 01 * * *  /usr/bin/tar cvf  /dev/rmt/0  /export/home/matt
```

View the results by running `crontab –l`

Remember to set the editor to `vi` (it makes life easier):

```
$ EDITOR=vi; export EDITOR
```

Disk configuration and naming

Describe the physical device names that are used to identify a system's devices

Physical names describe the full path name for a device in a device tree. All names are created under the `/devices` directory when they are first installed and recognized by the system.

This tree is a hierarchy of interconnected buses with the devices attached to the buses as nodes. The root node is the main physical address bus.

Each device node can have:

Properties - data structures describing the node and its associated devices

Methods - software procedures used to access the device

Data - initial values of the private data used by the methods

Children - other device nodes attached to the given device node. Nodes with children are usually buses while nodes without children are usually devices

Parent - the node above the given device node

The full device path name identifies a device in terms of its location in the device tree by identifying a series of node names separated by slashes with the root indicated by a leading slash. Each node name in the full device path name has the form:

`driver-name@unit-address:device arguments`

Where *driver-name* identifies the device name, *@unit-address* is the physical address of the device in the address space of the parent and *:device arguments* is used to define additional information regarding the device software.

Devices are referenced in three ways:

Physical device name (full device path name)

Logical device name

Instance name

Identify logical device names used by system administrators to reference disk devices and explain when they are used in the Solaris™ environment

Logical device names are used to identify disk, tape and CD-ROM devices and provide either raw access (one character at a time) or block access (via a buffer for accessing large blocks of data). The logical name of SCSI devices identifies the SCSI controller (bus), target (SCSI tap ID), drive (always 0, except when used in disk drive clusters) and slice (partition).

For example: **/dev/dsk/c1t2d0s3**

dsk identifies the device as a block disk (rdsk would indicate a raw disk) addressed as SCSI controller **1**, target **2** drive **0** and slice **3**.

Logical device names are located under the */dev* directory and are linked to the appropriate physical device name file under the */devices* directory.

Logical device names are used by the following commands:

`df` (block)

`fsck` (raw)

`mount` (block)

`newfs` (raw)

`prtvtoc` (block or raw)

Determine the type(s) of disk devices and disk device interfaces on your system using the format utility or dmesg command

The format command, used to manage disks, also prints *logical and physical names* of all attached devices.

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:

       0. c0t0d0 <ST38420A cyl 16706 alt 2 hd 16 sec 63>
          /pci@1f,0/pci@1,1/ide@3/dad@0,0


Specify disk (enter its number):
```

The `dmesg` command shows the instance and physical names of the disk devices the kernel knows about.

```
Oct  1 11:07:57 server12 unix: root nexus = Sun Ultra 5/10 UPA/PCI
(UltraSPARC-IIi 360MHz)
Oct  1 11:07:57 server12 unix: pci0 at root: UPA 0x1f 0x0
Oct  1 11:07:57 server12 unix: pci0 is /pci@1f,0
Oct  1 11:07:57 server12 unix: PCI-device: pci@1,1, simba0
Oct  1 11:07:57 server12 unix: PCI-device: pci@1, simba1
Oct  1 11:08:13 server12 unix: PCI-device: ide@3, uata0
Oct  1 11:08:13 server12 unix: uata0 is /pci@1f,0/pci@1,1/ide@3
Oct  1 11:08:13 server12 unix: dad0 at pci1095,6460
Oct  1 11:08:13 server12 unix: target 0 lun 0
Oct  1 11:08:13 server12 unix: dad0 is /pci@1f,0/pci@1,1/ide@3/dad@0,0
Oct  1 11:08:14 server12 unix: <ST38420A cyl 16706 alt 2 hd 16 sec 63>
Oct  1 11:08:14 server12 unix: root on
/pci@1f,0/pci@1,1/ide@3/disk@0,0:a fstype ufs
Oct  1 11:08:15 server12 unix: PCI-device: ebus@1, ebus0
Oct  1 11:08:15 server12 unix: su0 at ebus0: offset 14,3083f8
Oct  1 11:08:15 server12 unix: su0 is
/pci@1f,0/pci@1,1/ebus@1/su@14,3083f8
Oct  1 11:08:15 server12 unix: su1 at ebus0: offset 14,3062f8
Oct  1 11:08:15 server12 unix: su1 is
/pci@1f,0/pci@1,1/ebus@1/su@14,3062f8
```

Note the additional information that this command provides: interface controller number, address of device controller and logical unit number (LUN).

Identify the instance device name

The kernel abbreviates the name of every device it knows about. The instance device name is how the kernel recognizes the physical disk attached. It looks something like:

sd*n*

where sd = SCSI disk and <*n*> represents the number of the device on the SCSI bus.

Display system configuration information with the prtconf command

To view memory and peripheral configurations, use the `prtconf` command.

```
# prtconf
```

```
System Configuration:  Sun Microsystems  sun4u
Memory size: 64 Megabytes
System Peripherals (Software Nodes):

SUNW,Ultra-5_10
    packages (driver not attached)
        terminal-emulator (driver not attached)
        deblocker (driver not attached)
        obp-tftp (driver not attached)
        disk-label (driver not attached)
        SUNW,builtin-drivers (driver not attached)
        sun-keyboard (driver not attached)
        ufs-file-system (driver not attached)
    chosen (driver not attached)
    openprom (driver not attached)
        client-services (driver not attached)
    options, instance #0
    aliases (driver not attached)
    memory (driver not attached)
    virtual-memory (driver not attached)
    pci, instance #0
        pci, instance #0
            ebus, instance #0
                auxio (driver not attached)
                power, instance #0
                SUNW,pll (driver not attached)
                se, instance #0
                su, instance #0
                su, instance #1
                ecpp (driver not attached)
                fdthree, instance #0
                eeprom (driver not attached)
                flashprom (driver not attached)
                SUNW,CS4231 (driver not attached)
            network, instance #0
            SUNW,m64B, instance #0
            ide, instance #0
                disk (driver not attached)
                cdrom (driver not attached)
                dad, instance #0
                sd, instance #0
        pci, instance #1
    SUNW,UltraSPARC-IIi (driver not attached)
    pseudo, instance #0
```

Describe the function of the /etc/path_to_inst file

Instance names are bound to the full physical device names using the `/etc/path_to_inst` file. The kernel requires this file to keep track of all the devices attached to the system.

The file shows the physical name on the left side, and the instance number and device type on the right. Examine the output below. The numbers in bold are the instance numbers:

"/pci@1f,0" **0** "pci"
"/pci@1f,0/pci@1,1" **0** "simba"

```
"/pci@1f,0/pci@1,1/ebus@1" 0 "ebus"
"/pci@1f,0/pci@1,1/ebus@1/power@14,724000" 0 "power"
"/pci@1f,0/pci@1,1/ebus@1/fdthree@14,3023f0" 0 "fd"
"/pci@1f,0/pci@1,1/ebus@1/SUNW,CS4231@14,200000" 0 "audiocs"
"/pci@1f,0/pci@1,1/ebus@1/su@14,3062f8" 1 "su"
"/pci@1f,0/pci@1,1/ebus@1/se@14,400000" 0 "se"
"/pci@1f,0/pci@1,1/ebus@1/su@14,3083f8" 0 "su"
"/pci@1f,0/pci@1,1/ebus@1/ecpp@14,3043bc" 0 "ecpp"
"/pci@1f,0/pci@1,1/ide@3" 0 "uata"
"/pci@1f,0/pci@1,1/ide@3/sd@2,0" 0 "sd"
"/pci@1f,0/pci@1,1/ide@3/dad@0,0" 0 "dad"
"/pci@1f,0/pci@1,1/network@1,1" 0 "hme"
"/pci@1f,0/pci@1,1/SUNW,m64B@2" 0 "m64"
"/pci@1f,0/pci@1" 1 "simba"
"/options" 0 "options"
"/pseudo" 0 "pseudo"
```

Disk partitions and format

Define a disk label

A disk label acts as a table of contents for a physical disk. Another name for the disk label is Volume Table of Contents (VTOC). It is found on the first sector of the disk. The `format` command creates the disk label on a volume.

Define disk partitions/slices

A partition (a.k.a. slice) is a collection of disk cylinders grouped together as a single unit. Technically, the boundaries of a slice are defined by offset and size; i.e. the first partition begins at 0 and is 50 cylinders in length. The next partition is 'offset' 50 cylinders and begins at 51, etc.

Display a disk's volume table of contents with the prtvtoc command

Running `prtvtoc` generates output similar to the following:

```
* /dev/dsk/c0t0d0s0 partition map
*
* Dimensions:
*     512 bytes/sector
*      63 sectors/track
*      16 tracks/cylinder
*    1008 sectors/cylinder
*   16708 cylinders
*   16706 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
* Unallocated space:
*       First     Sector    Last
```

```
*       Sector     Count    Sector
*    15732864   1106784  16839647
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       0      2    00          0   2097648   2097647   /
       1      4    00    2097648   4195296   6292943   /usr
       2      5    00          0  16839648  16839647
       3      0    00    6292944   4195296  10488239   /export
       4      0    00   10488240   2097648  12585887   /opt
       5      7    00   12585888   2097648  14683535   /var
       6      3    01   14683536   1049328  15732863
```

Use the format utility to partition a disk

The `format` utility is an interactive program that assists in the configuration of attached disks. It allows a systems administrator to define custom partitions and commit them to a table ('label' the disk). It also provides functionality for finding and repairing disk defects.

Use the format utility to create and save a customized partition table

Once defined, partition schemes can be saved and recalled for later use (perhaps on similar disks). To save a partition scheme for reuse, 'name' the disk in the `partition>` part of format. Names must be in quotes. From the main `format>` menu, invoke 'save' to commit the partition definition to disk in the `/etc/format.dat` file.

To recall a saved partition scheme, invoke the 'select' option of the `partition>` menu. Select the number of the saved partition, and then 'label' the disk to make the setup active.

Introduction to file systems

Define the term file system

Simply defined, a file system is a collection of files and directories that organize information into manageable pieces. User data is kept separate from system data through the use of a hierarchical system.

Although there is a physical definition of data on a disk partition, file systems may also be distributed and therefore accessed over a network. The operating system manages this through standard control structures.

Define the contents of each of the standard Solaris™ 7 file systems

Solaris™ has support for disk-based, network-based, and RAM-based file systems.

Disk-based systems include ufs - the Solaris™ standard, hsfs - High Sierra file system (a.k.a. ISO 9660 for CDROMs) and pcfs - an attempt at reading and writing files from DOS systems.

Network-based includes NFS.

RAM-based file systems include support for RAM disks, or allocating space in memory for programs to run while the OS is running.

Create a new ufs file system

Disk partitions must have a file system on it before data can be written. In Solaris™, the `newfs` command is provided for this task. (`newfs` is actually a front-end for `mkfs`, a more powerful file system command).

```
# newfs /dev/rdisk/c0t0d0s4
```

will create a new ufs file system on partition 4.

Describe why fsck is necessary

`fsck` is needed to check for inconsistencies in the data being stored on a disk. It uses known parameters (like the link counter feature of an inode) to verify disk integrity. `fsck` ensures that data linked to by inodes exists and is unique. The ultimate goal is to protect against file system corruption.

Describe how to check and repair a file system

File systems in Solaris™ are checked using `fsck`. During startup, `fsck` runs in non-interactive mode and corrects basic file system inconsistencies. It enters into interactive mode when a systems administrator must make decisions about the suggested corrections.

By default `fsck` checks the file systems listed in the `/etc/vfstab` file that have an entry in the device to `fsck` field.

`fsck` cannot be run on a busy file system. The system must be in single-user mode, or the file system being checked must be unmounted, before the utility is run.

Display disk space usage by file systems

The `df` command shows information about mounted file systems.

```
df [-k] [directory]
```

Specifying the `–k` option shows the amount of available space on a file system, less the space occupied by the operating system.

```
# df –k
Filesystem            kbytes    used   avail capacity  Mounted on
/proc                      0       0       0     0%    /proc
/dev/dsk/c0t0d0s0    1015542  124619  829991    14%    /
/dev/dsk/c0t0d0s1    2052750  833705 1157463    42%    /usr
fd                         0       0       0     0%    /dev/fd
/dev/dsk/c0t0d0s5    1015542  291674  662936    31%    /var
/dev/dsk/c0t0d0s3    2052750  458708 1532460    24%    /export
/dev/dsk/c0t0d0s4    1015542  390887  563723    41%    /opt
swap                  378424     328  378096     1%    /tmp
```

Display the size of a directory

The du command lists the number of disk blocks in use by files and directories

`du [-a] [-s] [-k] [directory]`

Specifying the –k option forces the output to show in Kilobytes (it shows 512K disk bytes by default). –s provides summary information about a directory.

```
# du –k
521     ./Reader/help
689     ./Reader/res
10      ./Reader/desktop/olwm
11      ./Reader/desktop
15      ./Reader/sparcsolaris/app-defaults
1161    ./Reader/sparcsolaris/bin
5321    ./Reader/sparcsolaris/lib
283     ./Reader/sparcsolaris/plug_ins
6783    ./Reader/sparcsolaris
8414    ./Reader
10      ./bin
1338    ./Fonts
70      ./Browsers/sparcsolaris
76      ./Browsers
19599   .
```

Display disk usage by user name

The quot command displays disk space in Kilobytes used by users.

`quot [-af] [file system]`

Use quot –af to display total disk usage and the number of files by each user on all mounted file systems.

```
...
/dev/rdsk/c0t0d0s3:
456105   7128   root
 2588     104   #1009
    5       5   joebob
    4       4   josephh
    2       2   mattk
    2       2   test5
    1       1   jjaballa
    1       1   lanman
```

Mounting file systems

Mount and unmount local file systems

Mounting file systems refers to the attachment of separate file systems to the existing file system tree. New file systems join the tree at empty directories known as mount points.

File systems listed in the `/etc/vfstab` file are mounted automatically at boot-time. Additional file systems may be mounted using the mount command.

`mount` by itself shows what local file systems are mounted.

```
# mount
/proc on /proc read/write/setuid on Sun Sep 17 23:20:54 2000
/ on /dev/dsk/c0t0d0s0 read/write/setuid/largefiles on Sun Sep 17
23:20:54 2000
/usr on /dev/dsk/c0t0d0s1 read/write/setuid/largefiles on Sun Sep 17
23:20:54 2000
/dev/fd on fd read/write/setuid on Sun Sep 17 23:20:54 2000
/var on /dev/dsk/c0t0d0s5 read/write/setuid/largefiles on Sun Sep 17
23:20:54 2000
/export on /dev/dsk/c0t0d0s3 read/write/setuid/largefiles on Sun Sep 17
23:21:13 2000
/opt on /dev/dsk/c0t0d0s4 read/write/setuid/largefiles on Sun Sep 17
23:21:13 2000

/tmp on swap read/write/setuid on Sun Sep 17 23:21:13 2000
```

```
# mount /dev/disk/c0t0d0s7 /mnt
```

will mount slice seven of the disk at the directory `/mnt`

```
# umount /mnt
```

will unmount mounted file systems at the specified mount point

Two additional commands, `mountall` and `unmountall`, are also available to mount multiple file systems at once. They both have two options, `-r` and `–l`. Specifying `–l` will mount local file systems in the `/etc/vfstab` file. Specifying `–r` will mount remote file systems specified in the `/etc/vfstab` file.

Mount a file system of a specified file system type

To specify the type of file system to be mounted, use the –F option.

```
# mount –F pcfs /dev/floppy /mnt
```

will mount a PC-formatted floppy at `/mnt`. When `–F` is not included, ufs is the default.

Mount a file system that disables the default largefiles option

By default, Solaris™ 7 now supports files that exceed 2GB each on file systems. The file system must be forced to limit files written to smaller than 2GB by using the `nolargefiles` option. Use a command similar to the following to accomplish this:

```
# mount –o nolargefiles /dev/dsk/c0t0d0s7 /mnt
```

Set up your system to mount a local file system automatically at boot time

To mount a local file system at boot time, create an entry for it in the `/etc/vfstab` file. An example of the `/etc/vfstab` file:

```
#device         device         mount        FS     fsck    mount   mount
#to mount       to fsck        point        type   pass    at boot options
#
#
/dev/dsk/c1d0s2    /dev/rdsk/c1d0s2 /usr   ufs     1       yes    -
fd      -          /dev/fd          fd      -       no      -
/proc   -          /proc            proc    -       no      -
/dev/dsk/c0t0d0s6   -      -        swap    -       no      -
/dev/dsk/c0t0d0s0  /dev/rdsk/c0t0d0s0      /       ufs     1      no -
/dev/dsk/c0t0d0s1  /dev/rdsk/c0t0d0s1      /usr    ufs     1      no -
/dev/dsk/c0t0d0s5  /dev/rdsk/c0t0d0s5      /var    ufs     1      no -
/dev/dsk/c0t0d0s3  /dev/rdsk/c0t0d0s3      /export ufs     2      yes -
/dev/dsk/c0t0d0s4  /dev/rdsk/c0t0d0s4      /opt    ufs     2      yes -
swap                -      /tmp     tmpfs   -             yes     -
```

Remember the field labels! Order matters!

Backup and recovery

Dump a file system to tape using the ufsdump utility

`ufsdump` backs up a file system. It can perform a full backup or an incremental backup depending on the options specified.

```
ufsdump options [ arguments ] files_to_backup
```

Options include a numerical dump level, 0 – 9, where 0 is a full backup and 1 – 9 are incremental levels. The default location for backup is `/dev/rmt/0` unless a different one is specified using the `f` argument. For example, specifying:

```
# ufsdump 2v /dev/rdsk/c0t0d0s0
```

will backup to tape and verify everything that has changed since the last 0 backup was performed.

```
# ufsfump 3v /dev/rdsk/c0t0d0s0
```

will backup to tape and verify everything that has changed since the last 2 backup was performed.

Restore files or a file system from tape using the ufsrestore utility

`ufsrestore` retrieves the files stored using `ufsdump`.

```
ufsrestore options [ arguments ] [ files_to_restore ]
```

Options include `i` for interactive mode, `r` for entire restore, `t` for viewing the table of contents, v for viewing the names of the restored files.

```
# ufsrestore t
```

will generate a listing of the files stored on `/dev/rmt/0`

An interactive restore would be performed using:

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

Recover the root (/) or /usr file systems

The root file system must be recovered using `ufsrestore` after booting from a CDROM. The following steps can be performed:

```
ok boot cdrom –s
# newfs /dev/rdsk/c0t0d0s0
# mount /dev/dsk/c0t0d0s0 /restore
# cd /restore
# ufsrestore rvf /dev/rmt/0
# installboot bootblk /dev/rdsk/c0t0d0s0
# fsck /dev/rdsk/c0t0d0s0
# init 6
```

Backup and restore a directory using the tar utility

`tar` is a command to backup individual files and directories.

```
tar options [ arguments ] files_to_tar
```

Options include c to create, t to list table of contents, f to specify a filename, v to print names as they are processed, or p to restore original permissions.

```
# tar cvf matt.tar *
```

will create a tar file called matt.tar with all the files in the local directory.

```
# tar xvf matt.tar
```

will restore the files from matt.tar to the current directory

Position a tape to a selected data set using the mt utility

`mt` is used to directly manipulate a tape device.

```
mt [ -f tape_device ] command [ count ]
```

```
# mt –f /dev/rmt/0 fsf 5
```

would fast-forward the tape device past the first 5 data sets stored.

The LP print service

List the operating systems supported by the Solaris™ print service

Solaris™ supports printing as defined in the BSD printing protocol RFC 1179. This includes Solaris™ 1.x/2.x, HPUX, AIX, Windows™ NT and Novell.

Describe the functions of the line printer (LP) service

There are 5 major features of the LP service to remember:

Queuing – jobs are processed in the order in which they are received

Tracking – jobs are managed throughout the print process, and resumed in case of errors

Fault Notification – error messages are logged or displayed on the system console

Initialization – printers are initialized before a job is sent

Filtering – data is converted to formats printers understand, like postscript or PCL

Describe what a print server and print client are

A print server is a computer configured to send jobs to a locally attached or networked print device that it knows about. The `lp` daemon running on the server maintains a print queue (also called a spool) to store jobs that are ready to print.

A print client is any workstation or server configured to access a remotely shared printer, via a print server.

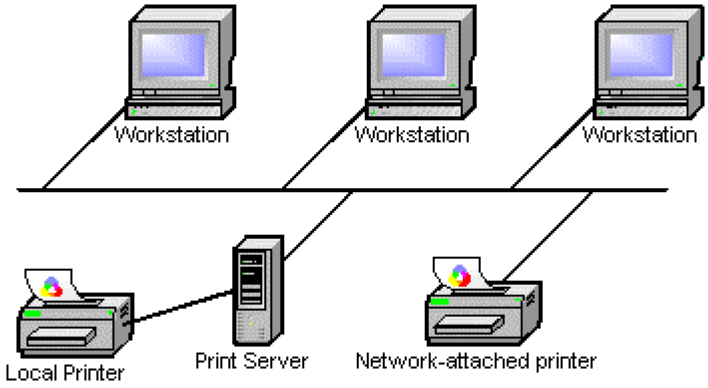
Define the terms local printer and remote printer

A local printer is one attached to the computer itself via a physical cable. It may also be a network-attached printer that is shared by a print server.

A remote printer is one that a user accesses via a print server (over a network).

Note: a network-attached printer is a device that supports printing without being connected to a physical computer. It usually has an IP address and hostname.

Diagram local and remote print models



Verify that a printer type exists in the terminfo database

View the contents of the `/usr/share/lib/terminfo` subdirectories. Look for a directory that has an abbreviated name for the type of printer that is being installed. Printer models are arranged alphabetically by directory.

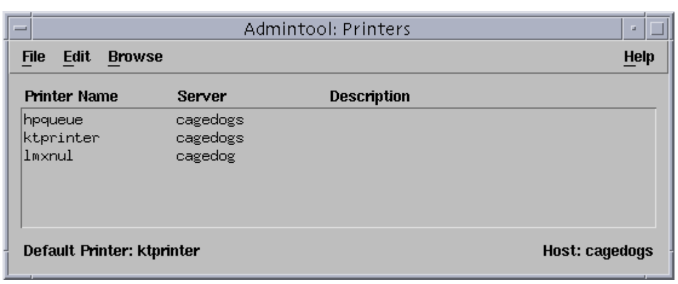
```
# ls /usr/share/lib/terminfo/o

o31        oc100      ofos       one_line     osborne1   ovi300
oadm31     oconcept   ojerq      one_linepty  otty5410   owl
oblit      ofortune   omron      osborne      otty5420   ozzie
```

If the printer is not in the `terminfo` database, refer to the Sun guide *User Accounts, Printers, and Mail Administration* to find out how to add one.

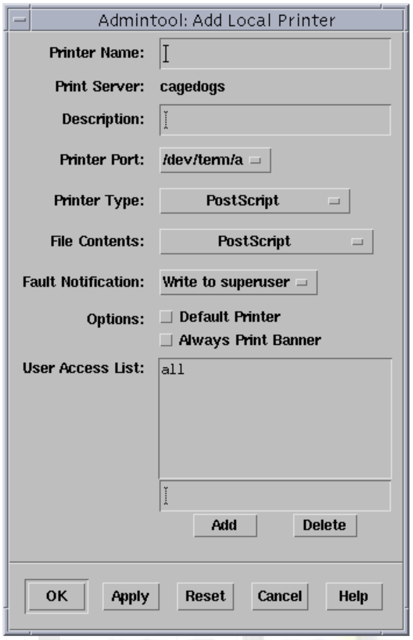
Use Admintool™ to add a local and remote printer to a system

Start `admintool` and select printers from the browse menu.

In the printers window, choose to Add a local printer from the Edit menu:

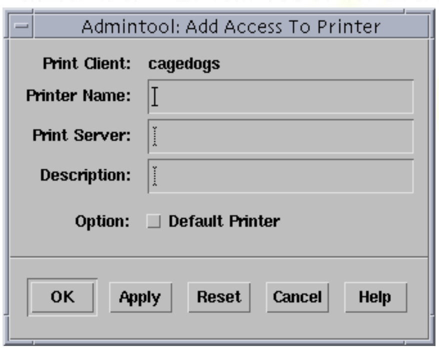


Specify the printer type and a port to use:



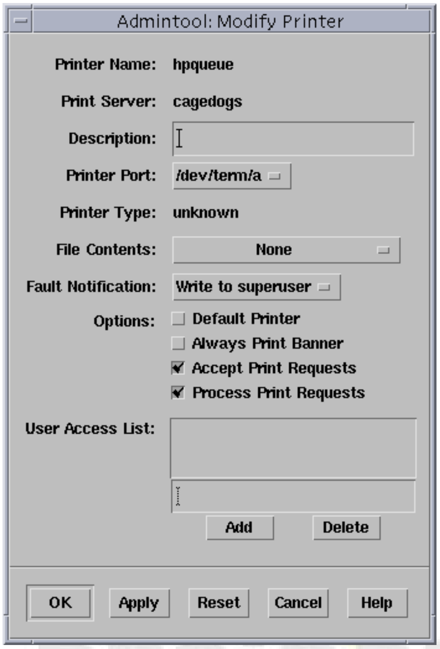
For a remote printer, choose to Access a printer from the Edit menu.

Enter in the printer (spooler) name and print server name.



Modify the configuration of a printer using Admintool™

Select a local printer from the printer menu, and then select Modify from the Edit menu.



Print Commands

Use the lp command to print files

`lp` queues data for printing.

```
lp [ -options ] filename
```

```
$ lp cramsession.txt
```

Use the lpstat command to monitor print jobs

Use `lpstat` to monitor print queues.

```
lpstat [ -options ]
```

Options include `–d` to display the default printer, `-o` which displays the status of requests on all printers, and `–t` which displays complete status information for all printers, including active queued jobs.

```
$ lpstat

hs-printer10          root               1098   Sep 13 05:45 on pserver
hs-printer11          root               1098   Sep 13 05:46

$ lpstat –d
system default destination: ktprinter
```

Use the cancel command to cancel print jobs

`cancel` will stop queued jobs, or even the actively printing job.

```
cancel [ job_name ] [printer]
```

```
$ cancel hs-printer12
```

will cancel the printing of cramsession.txt on the active printer (assuming that hs-printer12 is the print job name associated with printing the cramsession.txt file).

Use lpadmin to set up a printer class

A printer class is a group of printers (organized by type, location or function) that share the work of jobs sent to a single queue name.

A class is created when the first printer is added to it. Use the –c switch with lpadmin to create a printer class

```
# lpadmin –p printer1 –c marketing-department
```

Will create a printer class marketing-department and add printer1 to it.

Manually designate a default printer destination using the lpadmin command or the LPDEST environment variable

```
# lpadmin –d hs-printer1
```

will designate the printer hs-printer1 to be the default

In the Bourne or Korn shell, the LPDEST variable is set:

```
$ LPDEST=hs-printer1; export LPDEST
```

The c-shell requires:

```
workstation% setenv LPDEST hs-printer1
```

Use the lpmove command to move a queued print request from one printer to another

Use lpmove only when the original printer has stopped accepting jobs. To move jobs from hs-printer1 to hs-printer2:

```
# reject –r "printer is down" hs-printer1
# lpmove hs-printer1 hs-printer2
```

when hs-printer1 is ready for service:

```
# accept hs-printer1
```

Assign priorities to print requests and move a job to the top of the queue

Solaris™ assigns a default priority of 20 to each print job. Lower numbers have higher priorities. The range is 0 – 39. The –d option of lp will change the priority. For a large job, it is a good idea to lower the priority:

```
$ lp –d hs-printer1 –q 30 megaprint.txt
```

To move a job to the front of the queue, assign it priority 0.

```
$ lp –d hs-printer1 –q 0 rightnow.txt
```

Stop and start the the LP print service

```
# /etc/init.d/lp stop
# /etc/init.d/lp start
```

Solaris™ and Jumpstart™ are registered trademarks of Sun Microsystems, Inc.

Special Thanks to Matthew Kortas for contributing this Cramsession. Make sure to visit his site at: http://acm.cse.msu.edu/~kortasma