Sun Solaris 8 Certified Systems Administration I

Version 3.0.0

Sun Solaris 8 Certified Systems Administration I

Version 3.0.0

Abstract:

This study guide will help you to prepare for Sun exam 310-011, Solaris 8 System Administration I. Exam topics include simple file system concepts, the vi editor, OpenBoot PROM, system and file security, boot and run levels, adding and removing devices, and interacting with standard utilities.

Find even more help here:

Contents:

System Concepts

Match selected system administration terms to their respective definitions: daemons, shell, file system, kernel, operating system

Daemon - A daemon is a program that resides in system memory. When called upon it performs a specific system function. Administrators can specify which daemons they wish their system to run using scripts or start them manually from the command line. Finding out what daemons it has loaded in memory can sometimes identify a machine's purpose.

- **Shell** – The shell interprets and translates commands entered by the user into actions performed by the system. There are six by default in Solaris™ 8: Bourne Shell, Korn Shell, C Shell, Z Shell, TC Shell and BASH 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.
- **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.
- **Operating System** - An operating system is the intermediary between the users and the computer. It accepts user input, routes it to the proper device for processing, and returns program output back to the user. The operating system interacts with peripherals through device drivers written specially for each component of the system.

Define the effect of using various main command options when viewing online manual pages

Man pages (short for manual pages) provide online assistance for almost all UNIX commands on the system. Checking these files for help on arguments and switches, as well as sample commands, provides good background for the proper use of a command.

```
# man ls
```

will display a specially formatted (using nroff) text file that has information about `ls`. If the man page takes up more than one screen, the space bar will continue listing the file one screen at a time. Conversely, pressing the '`b`' key will back the display up one screen. Pressing `<enter>` will display the file one line at a time. The "`/`" will allow a search string to be entered, and the file will be searched for the keyword. Pressing '`n`' once a keyword is found will cause the man to continue searching for the string in the file until the next occurrence.
Typing '`h`' will display all the commands that can be used while viewing man pages.

The Boot Prom

State or recognize the combination of actions required to interrupt a non-responsive system

Should Solaris™ stop responding to user input, there is a way to get control of the system back. Pressing `stop-a` (the stop key at the left of the keyboard) will interrupt the running operating system and return to the OpenBoot OK prompt. The openboot command reset should be run at the ok prompt before any diagnostic tools are executed.
Pressing the `stop-n` key sequence while the system is booting will reset the values of the NVRAM to the factory defaults
And pressing `stop-d` will run the diagnostic mode (equivalent to `diag-switch?=true`) as the system boots up.

State the command strings used to manipulate custom device aliases

Solaris™ 8 can be booted from an external source. However, there may not be 'aliases' defined to tell the operating system about the new boot device. Creating a custom device alias using nvalias will correct this issue.
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.
Custom aliases can also be removed using the command nvunalias.

```
ok nvunalias <custom name>
ok setenv boot-device disk
ok reset
```

Installation

Describe the sequence of steps required to perform the Solaris 8 Operating Environment software installation on a networked standalone system

To install Solaris™ 8 onto a system, the computer requires a SPARC- or Intel-based system with 64 Megabytes of RAM, 2.3 Gigabytes of hard drive space, and access to a CDROM device.
The Solaris™ 8 operating system comes on two CDROM disks. There are several default clusters that can be installed initially:

Core – the base install, containing drivers - `SUNWCreq` - 718 Mb
End User – Core + OpenWindows and CDE - `SUNWCuser` - 1.2 GB
Developer – End User + compiler tools and man pages - `SUNWCprog` - 1.5 GB
Entire Distribution – All of Solaris™ 8 - `SUNWCall` - 1.9 GB
Entire Distribution plus OEM packages - `SUNWCXall` – 2.1 GB

There are two different install methods for at standalone system:

1. Command Line (CLI) - which defaults if there is no framebuffer installed
2. Graphical (GUI) - requires a framebuffer and is a little slower.

An administrator should have several pieces of information ready before performing an install:

- Hostname
- IP Address
- Name Service
- Subnet mask
- Geographic Region
- Root Password
- Language Support

An idea for a partitioning scheme for the disk; default systems use:
Root - Slice 0
Swap - Slice 1
`/export/home` - Slice 7

A good sample installation for a 4GB disk might look like:

```
/           0     300MB
/swap       1     2xRAM
/           2     OVERLAY
/var        3     500MB
/opt        4     500MB
/home       5     500MB
```

```
/usr         6      1400GB
/export      7      300MB
```

Identify the function of the following package administration commands: pkgadd, pkginfo, pkgchk, and pkgrm

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_8/Product SUNWaudio`
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_8/Product`
Example: `pkginfo -l SUNWman`

```
PKGINST:  SUNWman
      NAME:  On-Line Manual Pages
  CATEGORY:  system
      ARCH:  sparc
   VERSION:  41.0,REV=31
   BASEDIR:  /usr
    VENDOR:  Sun Microsystems, Inc.
      DESC:  System Reference Manual Pages
    PSTAMP:  tinkertoym09133331
  INSTDATE:  Nov 14 2000 15:23
   HOTLINE:  Please contact your local service provider
    STATUS:  completely installed
     FILES:    6420 installed pathnames
                  3 shared pathnames
                 74 directories
               73925 blocks used (approx)
```

Shows useful information about the package. Use `pkginfo | wc -l` to determine how many packages are installed on the system.

The `pkgchk` command is used to check completeness of the installed package
`pkgchk [ options ] [ -p path ] [ package_name ]`
Example: `pkgchk SUNWaudio`
(Note: the `pkgchk` does NOT display any output if the package is OK)
Example: `pkgchk -p /etc/shadow`
will check compare the shadow file to a checksum calculated during initial install to tell the administrator if the package (or file) has changed at all.

Use the `pkgrm` command to remove an installed package.
`pkgrm package_name`
Example: `# pkgrm SUNWaudio`

Identify the steps required to install a patch, verify which patches are currently installed, and remove a patch using the patchadd, patchrm, or showrev commands

On a Solaris™ 8 system, use the `patchadd -p` command to view installed patches. A legacy command, `showrev -p`, will also display the same information.
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
Use the command `patchrm <patch_name>` to remove an installed patch from the system. All files modified by the patch are removed unless:

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

In Solaris™ 8, `pkginfo`, `pkgadd` and `pkgrm` are now part of the admintool™ utility.

Initialization and Shutdown

Match the Solaris run levels to their intended functions

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 are running 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.

State the function of the following files or directories and the relationships between them: /etc/inittab, /etc/init.d, /etc/rc#(where # falls in the range of 0 to 6, or S), or /etc/rc#.d(where # falls in the range of 0 to 6, or S)

/etc/inittab

`/etc/inittab` contains information about actions to occur at each system run level, and is read by the init process. The three items it defines are:

1. Default run level
2. A list of processes to start/restart and monitor
3. What to do when a new run level is started

The lines of the `inittab` file look like:
`id:rstate:action:process`
where `id` is an up to 4 character identifier, `rstate` is the run level (or run levels) for the line to apply to, `action` is how the process will execute, and `process` is what gets executed.

```
s6:6:wait:/sbin/rc6           >/dev/msglog 2<>/dev/msglog </dev/console
fw:0:wait:/sbin/uadmin 2 0    >/dev/msglog 2<>/dev/msglog </dev/console
sc:234:respawn:/usr/lib/saf/sac -t 300
co:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` console login:
" -T sun -d /dev/console -l console -m ldterm,ttcompat
```

Some action keywords:
`initdefault`: specifies the default runlevel
`sysinit`: causes the line to execute and wait for completion before the file continues, and before the console prompt appears
`wait`: also causes the line to execute before the script continues
`respawn`: will start or restart processes as needed, or take no action if they don't exist
`powerfail`: executes the specified process if the powerfail signal is received
One interesting behavior is if the `rstate` field is left empty, the system will default to 0123456 and the system will reboot over and over again (the last runlevel is 6, remember).
Messages from scripts run from this file are saved to the `/dev/msglog` file.

/etc/init.d

`/etc/init.d` is the directory where run control scripts are stored. Each file is linked to the appropriate control script in the `/etc/rc#.d` directory.

The nice thing about having run control scripts linked to certain run levels is that, in the event a process needs to be controlled, and administrator can run each script individually. This way run levels do not need to be changed all of the time.

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

/sbin/rc#

`/sbin/rc#` is the script that runs scripts located in the appropriate `/etc/rc#.d` directory for a particular run level.

rc scripts	Function performed
`/sbin/rc0`	Runs scripts in `/etc/rc0.d/K*` to terminate system processes and daemons
`/sbin/rc1`	Runs the scripts in `/etc/rc1.d` to stop all system processes and daemons
`/sbin/rc2`	Runs the scripts in `/etc/rc2.d` to mount local file systems, start networking processes, clear the `/tmp` file system and start most of the system daemons
`/sbin/rc3`	Runs the scripts in `/etc/rc3.d` to share the contents of `/etc/dfs/sharetab` and starts NFS daemons
`/sbin/rc5` `/sbin/rc6`	Runs the `/etc/rc0.d/K` scripts to kill all processes and unmount file systems
`/sbin/rcS`	Runs the scripts in `/etc/rcS.d` to start the system in single user mode, which basically brings up the network, runs `fsck` and mounts local file systems and rebuilds the device tree (when new hardware is added).

/etc/rc#.d

A directory that contains links to the control scripts that start system processes at each run level. Scripts in the directory that start with 'S' are run when the system starts that run level. Scripts that start with a 'K' are run when the system exits that run level. Anything lowercase (s or k, or anything *OTHER* than S or K) is ignored.

Identify the commands used to change the run level of a system to a specified state

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

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>`

User Administration

Identify the following login procedures: logging into a system, logging out of a system, and changing login passwords

Logging into a UNIX system requires a username and password to already be defined for the user. A system administrator is in charge of creating and maintaining system accounts. To log into a system, enter the username at the `login:` prompt, followed by the correct, case-sensitive password. For security reasons, the password will not show when typed.
The user environment is comprised of a command shell where system commands are typed and output is received. To end a user session, type `exit` at the system prompt, or used the control sequence `CNTL-D` (`^D`).
To change a password, enter the command `passwd` at the command prompt. The system will prompt for the new password twice to avoid misspelling errors.

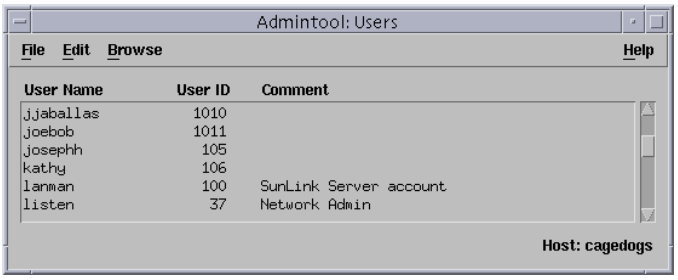
State the command used to identify which users are currently logged into the system

The `who` command is used to determine which users are logged into a system.
`who` has several flags that determine how users are logged in:
`-b` – the time and date of the last system reboot
`-H` – print header information in the who display
`-t` – when the system clock was last changed
`-u` – who is presently logged in
`-q` – displays a list of the current users (with total)
Flags may be used together, for example, date and time of the last reboot:

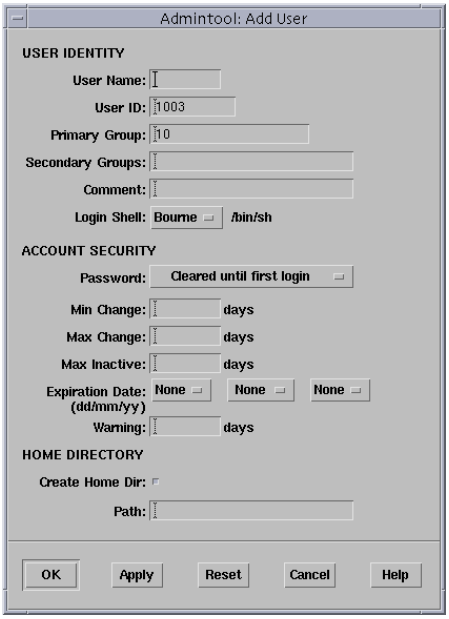
```
$ who -bH
NAME       LINE          TIME
  .       system boot   Oct  1 11:28
```

State the steps required to create user accounts on the local system using the admintool utility

`admintool`, a graphical administration tool, is provided with Solaris™ 8 to ease the management of necessary tasks on the system. `admintool` manages groups, hosts, printers, serial ports and software packages in addition to users.
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.

State the command syntax to add, modify, or delete user / group accounts on the local system with the useradd, groupadd, usermod, groupmod, userdel, or groupdel commands

Several command line programs exist to manage users on a system:
useradd - updates the `/etc/passwd` and `/etc/shadow` file, and copies files from `/etc/skel` to the new home directory.
`useradd [ -u uid ] [ -g gid ] [ -d directory ] [ -m ] [ -s shell ] [ -c comment ] name`
usermod - changes components of an existing user account.

```
usermod [ -uid ] [ -g group ] [ -d directory ] [-m ] [ -s shell ] [ -c
comment ] [ -l newlogonname ] [ -f inactive ] [ -e expire ] name
```

userdel - removes a users login account from the system. Can also remove home directories.

```
userdel [ -r ] name
```

groupadd - will create a new group on the system (in `/etc/group`).

```
grouadd [ -g gid ] [ -o ] groupname
```

groupmod - modifies an existing group.

```
groupmod [ -g gid  [ -o ] ] [ -n name ] groupname
```

groupdel - removes the group from the system (and the `/etc/group file`).

```
groupdel groupname
```

Given a user's login shell, list the shell initialization files used to set up a user's work environment at login

Each time a user logs into the system, the login shell reads several files to determine initial settings. `/etc/profile` is read to find system-wide settings. `/etc/.login` is read to define local environment settings. Initialization files define things like search path, environment variables, and window geometries.
Solaris™ 8 comes with 6 shells to choose from:

Shell Type	Shell Path	Login Init Files	New Shell started after login	System-wide init file
Bourne	/bin/sh	$HOME/.profile		/etc/profile
Korn	/bin/ksh	$HOME/.profile $HOME/.kshrc	$HOME/.kshrc	/etc/profile
C	/bin/csh	$HOME/.cshrc $HOME/.login	$HOME/.cshrc	/etc/.login
Z	/bin/zsh	$HOME/.zshenv $HOME/.zprofile $HOME/.zlogin	$HOME/.zshrc	/etc/zshenv /etc/zprofile /etc/zshrc /etc/zlogin
BASH	/bin/bash	$HOME/.bash_profile $HOME/.bash_login $HOME/.profile	$HOME/.bashrc	/etc/profile
TC	/bin/tcsh	$HOME/.cshrc or $HOME/.login	$HOME/.tcshrc or $HOME/.cshrc	/etc/csh.cshrc /etc/csh.login

Security

Identify how to search for regular expressions in the contents of one or more files

The `grep` utility is used to parse through existing files in search of matching strings. With its options, it can return output in a variety of ways.
Example: Finding all uses of a word

To find all uses of the word "Passing" (in any case) in the file scores.txt, and write with line numbers in the command's output:

```
# /usr/bin/grep -i -n Passing scores.txt
```

Example: Finding all empty lines

To find all empty lines in the standard input:

```
# /usr/bin/grep ^$
```

or

```
# /usr/bin/grep -v .
```

Example: Finding lines containing strings

Both of the following commands print all lines containing strings abc or def or both:

```
# grep -E 'abc def'

# grep -F 'abc def'
```

List command sequences used to display or modify file and directory permissions

File and directory permissions on files allow users to maintain control of files that they own, and prevent others users or processes from modifying or deleting them. A simple explanation of file permission's octal values is as follows: 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.

```
# ls -l
total 1
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile1
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile2
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile3
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile4
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile5
   0 -rw-r--r--   1 mkortas  p365            0 Jan 21 20:04 textfile6
   1 lrwxrwxrwx   1 mkortas  p365            9 Jan 21 20:04 textfile7 -
> textfile6
```

When a user wishes to change file and directory permissions, there are several utilities built into Solaris™ 8 to do so.
The `chown` tool is used to change ownership of files from one user to another user. The creator owns new files by default. The superuser uses the `chown` command to reassign the files ownership rights.

```
# chown username filename
```

The `chgrp` command is responsible for group permission changes. 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>
```

The programs `setuid` and `setgid` provide a mechanism for specifying permissions a file must use when being executed, instead of using the defaults that 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 he owns the file. `setgid` treats the execution as if the user belonged to the programs assigned group.
`setuid` and `setgid` show up in permissions mode listings as having an 's' in the execute field.
Example:
`-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
```

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
```

When the sticky bit is set on a directory, the directory may be publicly writeable, 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`.

Differentiate the effect of selected umask values on the permissions assigned to newly created files and directories

`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. Three digits represent umask. 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 command:

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

List in sequence the steps to create, modify, and delete access control lists (ACLs) on files

Access control lists provide a mechanism for finer control over the basic UNIX file permissions. They offer extended protection over and above what is provided by standard UNIX file permissions. A file has an acl set if its 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`

```
# getfacl

# file: newtext.txt
# owner: reader1
# group: users
user::rw-
user:reader1:r--        #effective:r--
group::r--              #effective:r--
mask:r--
other:r--
```

To remove an ACL from the file, run:

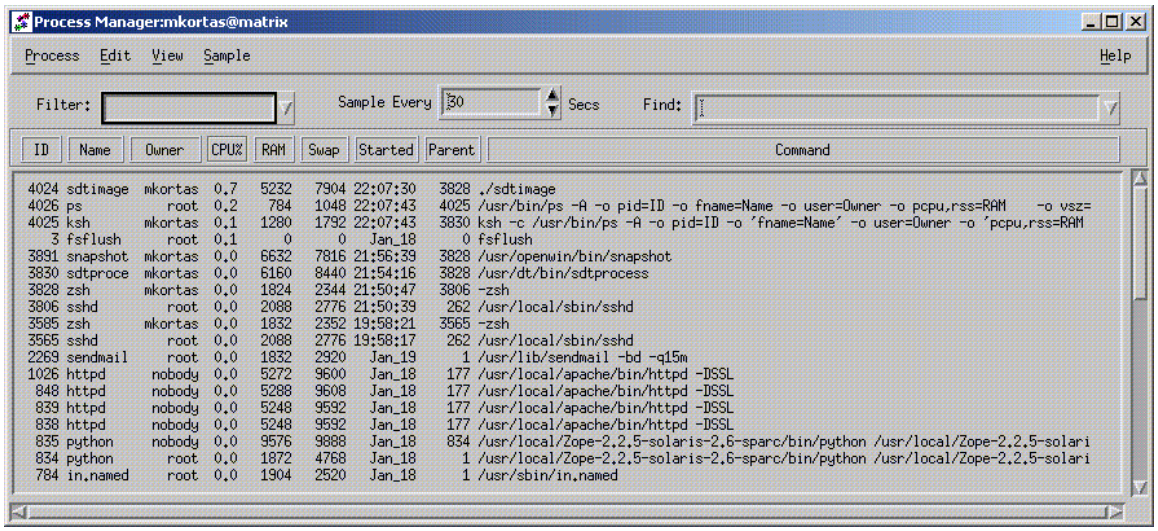
```
setfacl –d user:reader1:6 newtext.txt
```

Process Control

List the commands which display information for all active processes on the system

The most efficient way to list and manage processes on the system is using the `ps` command. The command will 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.
In Solaris™ 8, two new process control utilities are introduced: CDE Process Manager and `prstat`.
Process Manager is a GUI based tool to monitor and control processes.

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



`prstat` is an interactive way to monitor processes. It displays information sorted by CPU usage by default.

```
# prstat
PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  3637 kchiotis 1440K 1152K cpu0    58    0   0:00.00 0.1% prstat/1
   243 root       12M 3632K sleep   58    0   0:00.00 0.0% mysqld/6
   262 root     2512K 1384K sleep   50    0   0:00.13 0.0% sshd/1
```

```
  244 root     1736K 1280K sleep   58    0   0:00.00 0.0% ttymon/1
  200 root     2560K 2208K sleep   58    0   0:00.01 0.0% vold/6
  177 root     9320K 6608K sleep   59    0   0:00.00 0.0% httpd/1
  166 tacacs   1752K  992K sleep   48    0   0:00.00 0.0% tac_plus/1
  162 root     2648K 2048K sleep   53    0   0:00.00 0.0% nscd/9
  160 root     1872K  952K sleep   44    0   0:00.00 0.0% cron/1
 3585 kchiotis 2352K 1832K sleep   48    0   0:00.00 0.0% zsh/1
  784 root     2520K 1904K sleep   58    0   0:00.01 0.0% in.named/1
  151 root     2952K 1624K sleep   58    0   0:00.00 0.0% syslogd/8
  167 root     2592K 1248K sleep   59    0   0:00.00 0.0% ntpd/1
  147 root     1704K  696K sleep    0    0   0:00.00 0.0% inetd/1
  111 root     3024K 2064K sleep   59    0   0:00.00 0.0% ipmon/1
   52 root     2576K 1056K sleep    0    0   0:00.00 0.0% devfsadm/4
  134 nobody   2272K 1480K sleep   55    0   0:00.00 0.0% iplog/5
  190 root     1000K  688K sleep   59    0   0:00.00 0.0% utmpd/1
  233 root     1736K 1232K sleep   58    0   0:00.00 0.0% sac/1

Total: 34 processes, 81 lwps, load averages: 0.00, 0.00, 0.01
```

Some options for prstat are:

`-a`	displays separate user and process reports
`-c`	continuously prints new reports
`-p pids`	displays processes with pids that match the list
`-s key`	displays sorted list sorted by key. Keys can only be: cpu, time, size, rss and pri
`-t`	reports a usage summary for each user

State the effect of sending a specified signal to a process

Signals are notifications that are sent to processes. A signal interrupts whatever the process is doing and is handled immediately. Signals have integer values assigned to them, but are also associated with an alphanumeric name. To get a list of supported signals on Solaris™, type:

```
# kill -l
HUP INT QUIT ILL TRAP IOT EMT FPE KILL BUS SEGV SYS PIPE ALRM TERM USR1
USR2 CLD PWR WINCH URG POLL STOP TSTP CONT TTIN TTOU VTALRM PROF XCPU
XFSZ WAITING LWP FREEZE THAW CANCEL LOST RTMIN 39 40 41 42 43 44 RTMAX
```

A signal to a process can be generated using a control sequence (CNTRL-C, CNTRL-Z) or by using the `kill` command.

```
# kill -INT 2345
```

will send the interrupt (terminate) signal to process 2345

```
# kill -HUP 179
```

will send the hangup (reread init state) signal to process 179

```
# ^Z
```

will send the `SIGTSTP` signal to the current process, causing it to suspend.

List the commands used to terminate an active process

Use the `ps` command to obtain the PID of the process that needs to be terminated.
Run `kill`:
`# kill` <PID>
Use `pkill` to terminate the processes that contain specified strings. Example:
`# pkill –U root mail`
will kill all the processes owned by root with mail in the name.
Highlight the process and type the key combination `Control+C` in CDE Process Manager.

File Systems

List the different types of file systems in the Solaris Operating Environment

Solaris™ 8 officially supports three genres of file systems:

1. **Disk-based**
2. **Distributed**
3. **Psuedo**

Disk-based systems utilize physical medium, such as magnetic hard disks, CDROMs or DVD.

`ufs` - the standard UNIX disk system
`hsfs` - a special purpose file system used for CDROMS
`pcfs` - a file system that implements FAT32 for sharing of files with PC systems
`udf` - Universal Disk Format is a standard used with CDROM and DVD

Distributed file systems most often refer to remote disk-based file systems that are shared by other hosts.

`nfs` - remote file system sharing where the remote system looks as if it is local

Pseudo file systems reside in memory. They provide fast access to information needed by the kernel.

`tmpfs` - a storage space for temporary files that is destroyed at reboot
`swapfs` - the kernel stores virtual memory data here
`fsfs` - File Descriptor file system, which stores names for file descriptors
`procfs` - Process file system, which stores information about active system processes

State the effect of the commonly used options of the mount command

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
```

Mount does not require any special options to mount file systems of the default ufs type:

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

will mount slice seven of the disk at the directory `/mnt`
To specify the type of another 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.

Differentiate between the purpose of the /etc/mnttab and /etc/vfstab files

The `/etc/vfstab` file contains a list of those file systems which must be mounted at boot time. If a user mounts a file system using the mount command, and then the system is rebooted, the file system will not be available again until the mount command is rerun. An entry in `/etc/vfstab` will fix this.
`/etc/mnttab` contains a list of the currently mounted file systems, regardless of if they were mounted at boot time or not. Its contents looks similar to the output of just running the mount command with no options. The system modifies this file, which is actually a read-only file stored in the `mntfs` file system. No administration of this file is necessary.

Select correct statements about the intended purpose of the /etc, /opt, /usr, /export, and / (the root) directories

Certain directories / file systems have meaning to the Solaris™ file system.
`/etc/` is the directory where configuration files are stored
`/opt/` is a directory where third-party tools (optional packages) are installed
`/usr/` is the directory where system tools (binaries and libraries) are stored
`/export/` is the directory that can serve as the root of file systems that are shared
`/` (root) is the top of the directory hierarchy

List the steps required to access data on diskettes or CD-ROMs

Diskettes and CDROMs are considered removable media to the system. There is a process called volume management which handles the insertion and removal of file systems that aren't physically attached to the system.
Volume management is implemented through a process called `vold`. The process is started and stopped using:

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

`vold` automatically detects CDROMs that are inserted into the drive. It does not automatically detect diskettes until it is told to do so using the `volcheck` command.
The system mounts the removable file systems on `/floppy/floppy0` or `/cdrom/cdrom0`. The raw device paths to these devices are `/vol/dev/aliases/floppy0` or `/vol/dev/aliases/cdrom0`
There are two configuration files for volume management:
`/etc/vold.conf`
`/etc/rmmount.conf`

Files and Directories

State the commands used to reduce the size of files and directories for storage to tape

`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, `x` to extract files, `v` to print names as they are processed, or `p` 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
To compress files, Solaris™ 8 now includes the gzip utility. This program reduces the size of files to save space.

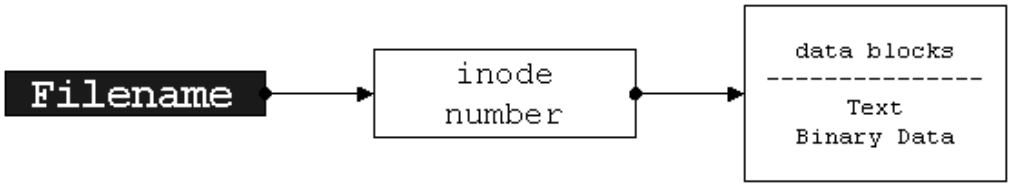
```
# gzip matt.tar
```
will compress matt.tar and rename the resulting (smaller) file `matt.tar.gz`

Match the file types of regular files, directories, symbolic links, device files, and hard links to their respective functions

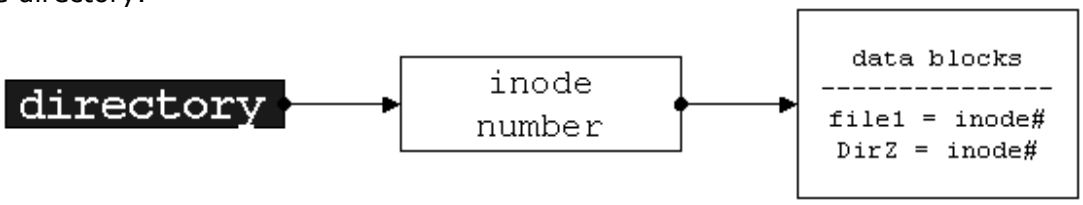
Solaris™ 8 supports all the standard file types that UNIX supports. The four main ones are:

1. **regular files** - identified by a "-" in the first field of an `ls` listing
2. **directories** - identified by a "d" in the first field of an `ls` listing
3. **symbolic links** - identified by a "l" in the first field of an `ls` listing
4. **device files** - identified by a "b" or "c" in the first field of an `ls` listing

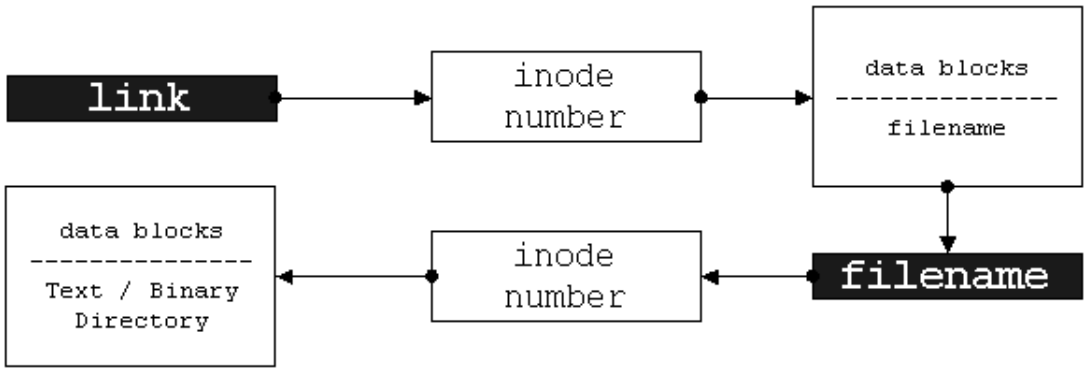
To understand file types, one must first understand the way records are stored on a disk. Solaris™ ufs makes use of a record type called an *inode*, which contains file information like size, permissions and ownership and pointers to the location of the actual data it contains. *Inode numbers* are generated when a new file system is created. It is a finite number, meaning that when all the inodes are used up, no more files can be created on the disk. The association between an inode and the filename is called a *hard link*.
Regular files hold data such as ASCII text, binary machine language, or image data. The file name is associated with an inode, which then points to the data blocks that contain file data.



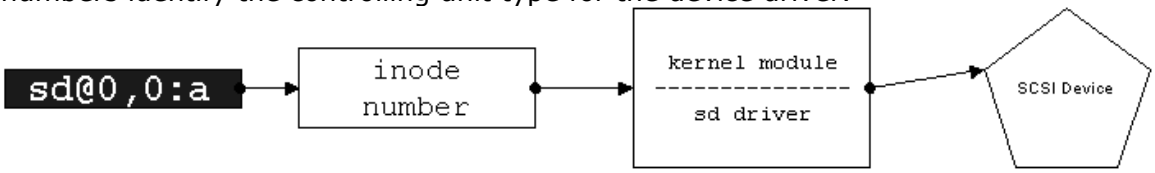
Directory files do not hold data, but rather point to a list of all files that exist within the directory.



Symbolic links are files that point to other files. The data the link contains is the pathname of the file it represents. Its size is based on the number of characters that make up the pathname to the original file.



Device files do not point to data blocks. Instead, their inode information refers to major and minor numbers. Major numbers describe a specific device driver. Minor numbers identify the controlling unit type for the device driver.



Device files can be one of two types, character or block. A character device means that I/O operations on the device are carried out at the smallest addressable unit (sector), which for disks is 512K. Block device means that I/O operations are carried out on the device based on a predefined block size.

The Boot Process

Match the boot command options to their respective functions

`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

Select the command that reports the current run level of a Solaris System

`who -r` is the command that will display the current runlevel.

```
   .       run-level 3  Jan 18 17:52     3      0  S
```

Given a sample run control directory, differentiate between the basic activity in a script whose name begins with an upper case S and a script whose name begins with an upper case K

The run control directory, in this example `/etc/rc2.d` contains symbolic links to startup scripts that are stored in `/etc/init.d`.
Scripts that begin with an "S" are automatically run when the system enters that runlevel; e.g., `S88sendmail` will execute the functionality of `/etc/init.d/sendmail start` when the system starts runlevel 3. The number 88 indicates the order in which S scripts in the directory are run.
Conversely, scripts that begin with a "K" are automatically run when the system exits that runlevel. These scripts are used to halt a process' activity in a clean fashion; i.e., script `K16apache` will stop the httpd process for apache before the system is shutdown.

Disk Configuration

Select the command used to add device configuration information for a new disk device without requiring a reboot of Solaris

Solaris™ 8 provides a new tool for adding devices to the system without requiring a reboot. This command is called `devfsadm`. The `devfsadm` tool works by attempting to load every system driver and attach to all device instances, eventually running across the new device on the system.
It is a good idea to restrict the `devfsadm` command to the particular type of device you are adding. This will save time because the system will not have to cycle through all devices if the type of new device is known beforehand.

```
# devfsadm -c device_class
```

Device classes can be: `disk`, `tape`, `port`, `audio`, or `pseudo`. You can specify multiple classes in the same command.
It can work at the driver level by specifying the -i option.

```
# devfsadm -i driver_name
```

Differentiate between the uses of a character (raw) disk (/dev/rdsk) and a block disk (/dev/dsk)

All disks on the system have a logical entry in the `/dev/rdsk` and `/dev/dsk` directories.
New devices that do not have file systems are communicated with using the raw device name. Utilities like `newfs` can write to the disk using the smallest addressable disk unit size as it writes a new file system.
Once the disk is attached to the system and formatted in ufs, it can be referenced as a block device. This is because the operating system is in control of the I/O to the

disk, and can mediate communications according to its agreed-upon block size (8KB is the ufs default).

Format

Identify the correct usage of the format command

`format` is a system administration tool for preparing a disk for the Solaris™ 8 operating system to use. `format` is run by the root user to divide the disk into slices, among other things. It is interactive and has nested menus for the available commands.
Once disk the slices are defined, `format` must 'label' the disk. The disk label, also known as the volume table of contents (VTOC), is stored on the first sector of the disk.

Select correct statements about the use of the menu selections for the format command

The format menu looks like the following:

```
   FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        save       - save new disk/partition definitions
        inquiry    - show vendor, product and revision
        volname    - set 8-character volume name
        !<cmd>     - execute <cmd>, then return
        quit
format>
```

The following menu selections are used when creating a new disk scheme:
partition - enters the partition submenu
label - writes the current partition scheme to the label
save - allows the admin to save the partition scheme to /etc/format.dat
verify - reads in and checks the disk label
quit

Select correct statements about the use of the menu selections for the partition subcommand under the format command

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

0 - 7 - allows the admin to change offset and size for the selected partition
select - allows the admin to pick a predefined scheme listed in /etc/format.dat
modify - make changes to the existing partition table
name - allows the admin to choose a name for the existing scheme
print - displays the current partition table
label - writes the current partition scheme to the disk label

Backup and Recovery

Match listed backup, archive, and restore utilities to their respective functional capabilities

`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 new one is specified using `f`. 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.
`# ufsdump 3v /dev/rdsk/c0t0d0s0`
will backup to tape and verify everything that has changed since the last 2 backup was performed.

`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, and `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
```

Identify the commands and steps required to backup a file system to tape

To back up the root filesystem, perform the following:

```
ok boot cdrom -s
# ufsdump 0uf /dev/rmt/0 /dev/rdsk/c0t0d0s0
# ufsrestore tvf /dev/rmt/0
```

Identify the commands and steps required to restore a file system from tape

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
# rm restoresymtable
# installboot bootblk /dev/rdsk/c0t0d0s0
# fsck /dev/rdsk/c0t0d0s0
# init 6
```

Basic Command Syntax

Using absolute or relative pathnames, select valid command strings to move between specified points within a given directory tree

All files and directories in the UNIX file system are ordered in a hierarchical fashion. Traversing the structure, also know as a tree, is accomplished with the command `cd` (change directory). The technical specification for the location of files (remember inodes?) was described above. For day-to-day use, it is more important to know the logical, hierarchical method for navigating the directory tree and locating files.
A file's location in relation to the entire file hierarchy is known as its absolute path. Specifying the absolute path will allow a user to specify the particular file from any other point in the tree. Example:

```
# ls –l /usr/local/lib/libz.so.1.1.3
```

will specify the library file located in `/usr/local/lib`.

If the user is currently in the /lib directory, changing to the `/usr/local/lib` directory is accomplished with:

```
# cd /usr/local/lib
```

Now that the absolute pathname is defined, the relative pathname is simply the location of the file in relation to the current location in the file tree. If the current working directory is `/usr/local`, the file `libz.so.1.1.3` can be specified:

```
# ls lib/libz.so.1.1.3
```

and the user can change to the directory:

```
# cd lib
```

Select the metacharacter combinations necessary to construct pathname abbreviations for access to files and directories within the directory tree

Metacharacters are also known as wildcards. They are symbols that act as replacements for existing characters and using them can save typing time as an administrator works with files and directories.

One of the most commonly used wildcards is * (star). It simply matches any character in a file or directory listing.

Examples:

```
# ls -l /etc/rc3.d/*
   6 -rwxr--r--   6 root     sys         2307 Sep  1  1998 /etc/rc3.d/S15nfs.server*
   2 -rwxr--r--   6 root     sys          404 Sep  1  1998 /etc/rc3.d/S76snmpdx*
   2 -rwxr--r--   6 root     sys          861 Sep  1  1998 /etc/rc3.d/S77dmi*
   4 -rw-r--r--   1 root     sys         1708 Sep  1  1998 /etc/rc3.d/README
# ls -l /lib/va*
  4 -rw-r--r--   1 bin      bin         1080 Sep  1  1998 /lib/values-Xa.o
  4 -rw-r--r--   1 bin      bin         1080 Sep  1  1998 /lib/values-Xc.o
  4 -rw-r--r--   1 bin      bin         1080 Sep  1  1998 /lib/values-Xs.o
  4 -rw-r--r--   1 bin      bin         1080 Sep  1  1998 /lib/values-Xt.o
  4 -rw-r--r--   1 bin      bin         1076 Sep  1  1998 /lib/values-xpg4.o
# cd /usr/share/rel*
# pwd
/usr/share/release_info
```

Another metacharacter is " (quotes). A set of quotes tells the shell to interpret what is in between the quotes as literal, and can be used in a situation where an administrator specifies a file name containing spaces:

```
# mkdir "Unix Software Repository"
# ls
Unix Software Repository
```

When working in a directory tree, it is sometimes useful to be able to return to the user's home directory quickly. The ~ (tilde) metacharacter specifies this:

```
# cd ~/textfiles
# pwd
/home/kchotis/textfiles
```

State the commands needed to list the contents of directories and determine the file types within a directory

Listing files in a directory is accomplished via the `ls` command. The `ls` command has many options for formatting the output of directory contents. For each file that is a directory, `ls` lists the contents of the directory; for each file that is an ordinary file, `ls` repeats its name. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.
In its most basic form:

```
# ls
total 18           2 bin@          2 logs@         2 requests/     2
temp@
   2 admins/       2 fifos/        2 model@        2 system/       2
tmp/
```

A more useful output is `ls -l`:

```
# ls -l
total 18
   2 drwxrwxr-x   2 lp       lp            512 Dec  6 10:07 admins/
   2 lrwxrwxrwx   1 root     root           23 Dec  6 10:07 bin ->
../../../usr/lib/lp/bin/
   2 drwxrwxr-x   4 lp       lp            512 Dec  6 10:19 fifos/
   2 lrwxrwxrwx   1 root     root           13 Dec  6 10:07 logs ->
../../lp/logs/
   2 lrwxrwxrwx   1 root     root           25 Dec  6 10:07 model ->
../../../usr/lib/lp/model/
   2 drwxrwxr-x   3 lp       lp            512 Dec  6 10:19 requests/
   2 drwxrwxr-x   2 lp       lp            512 Dec  6 10:07 system/
   2 lrwxrwxrwx   1 root     root           22 Dec  6 12:38 temp ->
/var/spool/lp/tmp/softcell/
   2 drwx------   4 lp       lp            512 Dec  6 10:19 tmp/
```

The mode printed by the `-l` option consists of ten characters (a combination of d, -, r, w, and x). The first character lists the type of file: `d` for directory, `-` for regular file, `c` for character device, `b` for block device, or `l` for symbolic link.

List the commands used to create or remove directories

The `mkdir` command creates directories with default permissions 777 (and altered by the umask value).

```
mkdir [ -m ] [ -p ] directory_name
# mkdir -m 700 example
```

will create a directory called example with permissions `rwx` for the owner.

```
# mkdir -p /tmp/mk/trash
```

will create the directory trash in `/tmp`, and will create `mk` if it does not exist already.

The `rmdir` command will destroy directory entries, providing that they are empty.

```
rmdir [ -ps ] directory_name
# rmdir example
```

will remove the `example` directory.

```
# rmdir -p example/widget
```

will remove the `widget` directory, and if `example` is now empty, remove `example` as well.

The command `rm -rf` is another way to recursively remove a directory. It recursively deletes every file in the directory and then removes the directory itself.

State the commands used to copy, create, rename, or remove files

The command `cp` copies files.

```
cp [ -r ] [ -i ] source destination
# cp cram1.txt cram2.txt
```

copies `cram1.txt` to `cram2.txt`.

```
# cp -r book1 old_book
```

will copy all the contents of the `book1` directory to the directory `old_book` recursively.

The command `mv` moves or renames files.

```
mv [ -f ] [ -i ] source destination
# mv cram1.txt cram2.txt
```

effectively renames the file `cram1.txt` to `cram2.txt`, because `cram1.txt` will no longer exist.

The command `rm` deletes files.

```
rm [ -f ] [ -i ] filename
# rm -f cram1.txt
```

will remove the file `cram1.txt` without prompting.

Note: specifying `-i` in any of the above commands will force an interactive mode that will check for a yes or no response before proceeding.

Editor

List the keyboard sequences that are required to switch between the three modes of operation used by the vi editor

vi is an standard text editor used for manipulating ASCII text files in UNIX. It is a very powerful tool with many features. However, because of its seemingly cryptic interface, some find it hard to transition into. It is important to remember that few system administrators can be successful without a perfunctory knowledge of vi.
vi has three modes of operation: command mode -- where program commands can be executed on blocks of characters, insert (edit) mode -- where text is actually entered into the file, and search mode -- where blocks of text can be queried.
From command mode, enter insert mode by pressing the i key. To exit insert mode, press the escape key. Search mode is entered by the / or ? key, and exited using the escape key. The escape key is used to toggle between command mode (the default) and all other modes.

Example session:

```
UNIX   ---> vi file --->  COMMAND  ---> i I a A o O --->  TEXT
SHELL  <---- ZZ <-------  MODE     <------ <Esc> <------  MODE
```

In command mode, typing ":" causes the editor to expect a command (shown on line 23 of the editor window). For example:
`:w` means 'write' the file to disk.
`:q` means quit out of the file.
Combining the two...
`:wq` means write and exit the file.
`:q!` means quit without saving changes
`ZZ` means quit and save changes

State the vi editor commands used to position and move the cursor, create and delete text, and copy or move text

The concept of vi centers on the use of the home row keys. All-important commands can be executed using standard keyboard keys, and no combinations of control or alt.
Movement:
- `h` - moves the cursor one postion left
- `j` - moves the cursor one line down
- `k` - moves the cursor one line up
- `l` - moves the cursor one position right
- `0` - moves the cursor to the beginning of the line
- `$` - moves the cursor to the end of the line
- `w` - moves the cursor forward one word

`b` - moves the cursor back one word
`G` - moves the cursor to the final line

Text Editing:
`a` - changes to insert/edit and allows new text to be appended
`A` - changes to insert/edit and appends to the beginning of line
`i` - changes to insert/edit and allows new text to be inserted
`I` - changes to insert/edit and inserts at end of line
`r` - replaces the character in the current position
`R` - replaces the character and all subsequent characters until escape
`x` - deletes the current character
`X` - deletes the character before the cursor
`dd` - deletes the current line
o – create new line below current line
0- create new line above current line

Copying Text:
`yy` - 'yanks' (copies) line(s) into the buffer for use with pasting
`p` - 'puts' (pastes) the stored line(s) onto the line below
`P` - 'puts' (pastes) the stored line(s) onto the line above

Moving Text: A user must specify move in command mode:
`5, 15 m 1` will move text from line 5 to 15 and place it starting at line number 1.

Match the correct vi command sequences with their respective search and replace functions

vi can search the entire file for a given string of text. A *string* is a sequence of characters. vi searches forward with the slash (/) key or backward with the question mark key (?). You execute the search by typing the command key, then *string* followed by RETURN. To cancel the search, press ESC instead of RETURN. You can search again by typing **n** (forward) or **N** (backward). Also, when vi reaches the end of the text, it continues searching from the beginning. This feature is called *wrapscan*.

vi supports a few special characters, which act as wildcards or search-exclusions. Note that XXXX stands for any number of characters; it could be **g**, **gefha**, or **23CG-4**. The special characters are: $. * [ ] ^ \.

`[XXXX]`	match any of the characters	Example:	`/sa[fn]`
`[X1-X2]`	match any characters between X1 and X2	Example:	`/[d-h]er`

vi can also search-and-replace, which means finding instances of a given string and replacing them with a new string. This search-and-replace operation is actually an ex command, and it has the following form: *line1, line2*s*/oldstring/newstring*

Remote Connection

State the command to perform remote system operations such as remote login, remote copy, and remote shell commands

There are three built-in methods for controlling remote access to a Solaris™ computer.

```
/etc/hosts.equiv
$HOME/.rhosts
/etc/ftpusers
```

The first two remote access control files are used to subvert the standard password-based mode of Solaris™. Their contents (if they exist) are checked before the password prompt appears to determine if the user has privileges on the system. Both files have a similar format for entries they contain:

```
hostname
hostname username
+
```

- A value for `hostname` indicates that all users from that particular host are trusted and may log into the system.
- A value for `hostname username` is a bit more restrictive, and means that only the username from a particular host is allowed access.
- A value of `+` means that all remote hosts on the network are trusted, and anyone may log in using rlogin without a password (assuming of course, they have a similarly named local user account)

`/etc/hosts.equiv` is a system-wide file, meaning that as long as a user has an account on the local host, and they are contacting the system from the hostname specified in the file, they will get access. The root user account is excluded from this list, and will always be asked for a password.

`$HOME/.rhosts` is a way for a specific user to use this functionality. If the `.rhosts` file is created in the home directory for a user and contains the name of a trusted host, they will not be prompted for a password upon logging into the system from that trusted host. Every user may use `.rhosts`, including the root user.

The access control described above controls the functionality of the commands `rlogin`, `rcp` and `rsh`.

rlogin is the program a user executes in their shell to log into a host where they are trusted without a password. Example:

```
# rlogin trusting_host
```

`rcp` is the program a user would use to copy a file from their system to a remote system on which they are trusted without a password. Example:

```
# rcp testfile remote_server:/tmp
```

`rsh` is the program a user runs to execute a shell command on a remote system on which they are trusted without a password. Example:

```
# rsh remote_host ls /tmp
```

Note: the above functionality is advertised as a good way to keep ASCII passwords from traversing a network. However, because of the uncontrolled nature of the `.rhosts` environments, allowing trusted access is a huge security risk. Remove the `/etc/hosts.equiv`, `$HOME/.rhosts`, and remove the shell and login lines from `/etc/inetd.conf` to disable this on a system.
Remote access might also come in the form of file transfers. To this end, the `/etc/ftpusers` file exists to restrict access to specific accounts that are not allowed to access the system via ftp. Any usernames on the system that appear in this file will be denied access should they connect via FTP.

State the subcommands that are used by the ftp utility to transfer files between a local system and a remote system

FTP stands for File Transfer Protocol, and is a standard UNIX method for transferring files from a system. FTP listens on port 21. The FTP daemon authenticates users using the `/etc/passwd` file. Some systems allow users to connect without entries listed in `/etc/passwd` -- these users are known as anonymous. Anonymous FTP is the *de facto* standard for sharing public software (or any software) on the Internet. Once a user connects to a system using FTP, they are able to traverse the directory tree they have access to as if they were on the local system. Commands like `cd` and `ls` work as they do on a local system.
When files are ready to be transferred, there are some additional commands that must be used.
`ascii` - specifies ASCII format, and is used to tell FTP that the file being sent or retrieved is in ASCII text format.
`binary` - specifies a binary format, which is all other file types besides ASCII.
`get` - instructs the remote computer to send a specified file.
`put` - instructs the remote computer to expect a specified file.
`mget` - 'multiple get' specifies more than one remote file to be sent. Used with wildcards.
`mput` - 'multiple put' specifies more than one local file to be sent. Used with wildcards.
`lcd` - changes directories on the local computer.
`prompt` - enables or disables confirmation prompts.
`hash` - enables or disables the printing of 'hash marks' that represents a block of bytes in a file and is a visual aid when files are actually being transferred.
An example FTP session might look like (typed commands are in bold):

```
# ftp somemachine.somewhere.com
This is a restricted system.  Unauthorized access is
 strictly prohibited.  All user activity is subject
 to monitoring and audit controls.  If you have a
 problem with this policy, please log off now.
```

```
220 somemachine.somewhere.com FTP server ready.
Name (localhost:kchiotis): ftp
331 Anonymous login ok, send your complete e-mail address as password.
Password:
230 Anonymous access granted, restrictions apply.
ftp> bi
200 Type set to I.
ftp> hash
Hash mark printing on (8192 bytes/hash mark).
ftp> prompt
Interactive mode off.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
pub
incoming
#
226 Transfer complete.
15 bytes received in 0.097 seconds (0.15 Kbytes/s)
ftp> cd /pub/solaris/freeware/8
250 CWD command successful.
ftp> mget gzip*
200 PORT command successful.
150 Opening BINARY mode data connection for gzip-1.2.4a-sol8-sparc-
local (365056 bytes).
######################
226 Transfer complete.
local: gzip-1.2.4a-sol8-sparc-local remote: gzip-1.2.4a-sol8-sparc-
local
365056 bytes received in 0.12 seconds (2862.90 Kbytes/s)
ftp> quit
```


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