SECTION 6. ESSENTIAL SOLARIS COMMANDS

This section provides an introduction to essential Solaris commands available to the System Administrator (SA) during a console session when logged in as root. The Solaris 2.x is sometimes referred to as SunOS 5.x; this section won't propagate that terminology. Online help is available in Solaris by typing man followed by the command or utility needed (see paragraph 6-12).

6.1 Purpose. This section provides a review of the more frequently used Solaris commands.

6.2 File/User/Manual Utilization Commands.

a. **su** - Become super-user or another user.

(1) *Description.* This command allows one to become another user without logging off. The default user name is root (super-user). To use **su**, the appropriate password must be supplied, unless the invoker is already root. If the password is correct, **su** creates a new shell process that has the real and effective user-ID, group-IDs, and supplementary group list set to those of the specified user name. To return to normal user-ID privileges, type an EOF character <Ctrl-d> to exit the new shell. All attempts to become another usr using **su** are logged in the log file (/var/adm/sulog).

(2) *Format.* # **su** [-] [username] [argument...]]

(3) *Examples.*

su -
Become super-user. You must know the super-user password.
su bin
Become user bin while retaining your previously exported environment.

(4) *Options.* None.

b. **pwd** - Print current working directory to the screen.

(1) *Description*. **pwd** writes an absolute path name of the current working directory to standard output (the monitor screen). If an error is detected, output will not be written to standard output. A diagnostic message will be written to standard error. "Cannot open .... " and "Read error in ..." indicate possible file system trouble and should

be referred to a Solaris SA. If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full path name to correct this situation.

(2) *Format*. # **pwd**

(3) *Example.* # **pwd**

(4) *Options.* None

c. **ls** - Lists the contents of a directory.

(1) *Description*. 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 and any other information requested. 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. There are three major listing formats. The default format for output directed to a terminal that is multi-column with entries sorted down the columns. The **-l** option allows single column output and **-m** enables stream output format. In order to determine output formats for the **-C**, **-x**, and **-m** options, **ls** uses an environment variable, COLUMNS, to determine the number of character positions available on one output line. If this variable is not set, the *terminfo* database is used to determine the number of columns, based on the environment variable *TERM*. If this information cannot be obtained, 80 columns are assumed.

(2) *Format.* # **ls** [-options] [file ...]

(3) *Example*. # **ls** - l /ajt/usr/ajt01

```
total 930
-rw-rw----    1    ajt01   sarss     135  Apr  1  1998   DBUALL.log
drwx------    2    ajt01   sarss     512  Apr 15 1998   Mail
-rw-rw-rw     1    ajt     sarss   53475  Apr 22 1998   PRTIS.out
```

The fields appearing on the screen using the long listing option are: total 930, which is the total disk space used in blocks; the file type and permissions, the number of links, user (owner) name, group name, file sizes in bytes, modification time, and file or directory name. The first line is a file where the owner has read and write (rw-) permissions, other members of the group have read and write (rw-) permissions and

others do not have permissions over the file (---); it has "1" link to it; the owner of the file is ajt01; the file belongs to the group "sarss"; it consists of 135 characters or bytes; it was last modified on April 1, 1998; and the name of the file is DBUALL.log. The second line is a directory where the owner has read, write and execute permissions (drwx------) for Mail. No other user can access this Mail directory. The mode printed under the **-l** option consists of ten characters. If the first entry is a dash (-) it is an ordinary file or the first character may be one of the following:

d - the entry is a directory;
D - the entry is a door;
l - the entry is a symbolic link;
b - the entry is a block special file;
c - the entry is a character special file;
p - the entry is a fifi (or "named pipe") special file;
s - the entry is an AF_UNIX address family socket;

The next nine (9) characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next set refers to permissions of others in the user-group of the file; and the last to all others. Within each set, the three characters indicate permission to read, to write, and to execute (rwx) the file as a program, respectively. For a directory, "execute" permission is interpreted to mean permission to search the directory for a specified file.

(4) *Options*.

-a List all entries, including those that begin with a dot.
-d If an argument is a directory, list only it's name.
-i Display the starting position for each directory and file.
-1 Print a long detailed listing.
-r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
-t Sort by time stamp (latest first) instead of by name.
-x Print multi-column output and sort entries across the page alphabetically.

d. **mkdir** - Make a new directory.

(1) *Description*. The **mkdir** command creates a named directory. It creates the named directory in mode 777 (full permissions) unless altered by the **-m** option. Standard entries in a directory (the files "." and "..") are made automatically. **mkdir** cannot create these entries by name. Creation of a directory requires write permission in the parent

directory. The owner-ID and group-ID of the new directory are set to the process's effective user-ID and group-ID, respectively.

(2) *Format*. # **mkdir** [-option] dir...

(3) *Examples*.

mkdir mydir
Creates a new directory name "mydir" in the current working directory.
mkdir #HOME/class
Creates a new directory named "class" in your HOME directory regardless of which directory you are currently in.
mkdir /ajt/usr/ajt01/newdir
Uses an absolute path name to create a new directory named "newdir" in your home directory (/ajt/usr/ajt01).
mkdir -m 700 mydir
Creates a new directory name "mydir" where the owner has read, write and execute (drwx------) permissions and no other user can access the directory.

(4) *Options*.

-m <u>mode</u> Allows you to specify the mode to be used for new directories.
-p With this option, **mkdir** creates the directory name by creating all the non existing parent directories first.

e. **cd** - Change directory.

(1) *Description*. If directory is not specified, your home directory (#HOME) is used as the new working directory.

(2) *Format*. # **cd** [directory]

(3) *Examples*.

cd /ajt
Changes directory to the subdirectory "ajt".
cd mydir
Uses relative pathname to change the current working directory to the subdirectory "mydir" in the user's HOME directory.
cd ..

Uses relative pathname to change the current working directory to the parent of the user's HOME directory, /ajt/usr/ajt01.

cd /ajt/usr/ajt01/class

Uses absolute pathname to change the current working directory to subdirectory "class" in the user's HOME directory.

cd

Changes the current working directory to the user's HOME directory.

(4) *Options.* None

f. **cp** - Copy files.

(1) *Description*. The **cp** command copies the complete contents of one file into another. It also allows you to copy one or more files from one directory into another providing you have proper permissions to do so. If you attempt to copy an existing file to another existing file the contents of the existing file will be lost. Use the **-i** option to prevent this. Solaris does not allow you to have two files with the same name in a directory. You can not copy directories with the **cp** command.

(2) *Format*. # **cp** [-options] source-file target-file

(3) *Examples*.

cp file1 file2

Copies the contents of file1 to file2.

cp ~/ajt/* /tmp

Copies a list of files to a destination directory.

(4) *Options*.

-i Interactive. Prompt for confirmation whenever the copy would overwrite an existing target-file. A "y" answer allows the copy to execute. Any other answer prevents **cp** from overwriting the target-file.

-r Recursive. Copy the directory and all its files, including any subdirectories and their files to the target-file.

g. **mv** - Move files.

(1) *Description*. Moving a directory and its files to a new directory with the <u>same parent directory</u> performs the function of renaming the directory. If the source and target

directory are on different file systems, **mv** copies the file and deletes the original; any links to other files are lost. After the move, the original file or directory is no longer addressable.

(2) *Format*. # **mv** [-options] source-file target-file

(3) *Examples*.

mv file1 file2
Moves file1 to file2.
mv -f yourdir mydir
In your HOME directory, moves directory "yourdir" to "mydir".

(4) *Options*.

-f Move the file without prompting even if it is writing over an existing target. This is the default if the standard input is not a terminal.

-i Prompt for confirmation whenever the move would overwrite an existing target. An affirmative answer means that the move should proceed. Any other answer prevents **mv** from overwriting the target.

h. **rm** - Removes a file.

(1) *Description*. The **rm** utility removes the directory entry specified by each file argument. If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with "y" (for yes), the file is deleted, otherwise the file remains. If the file is a symbolic link, the link will be removed, but the file or directory to which it refers will not be deleted. Users do not need write permissions to remove a symbolic link, provided they have write permissions in the directory. If multiple files are specified and removal of a file fails for any reason, **rm** will write a diagnostic message to standard error, do nothing more to the current file, and go on to any remaining files. If the standard input is not a terminal, the utility will operate as if the **-f** option is in effect.

(2) *Format*. # **rm** [-options] file ...

(3) *Examples*.

rm -i datafile

Prompt for confirmation before removing the file "datafile".
rm -f mydir
Remove all files in the directory "mydir" without prompting the user.

(4) *Options*.

-f Remove all files (whether write-protected or not) in a directory without prompting the user.

-i Prompt for confirmation before removable. The -f option will be ignored.

-r Recursively remove directories and sub-directories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting if the -f option is used or if the standard input is not a terminal and the -i option is not used. If the removal of a non-empty, write-protected directory is attempted, the utility will always fail, even if the -f option is used, resulting in an error message.

i. **rmdir** - Removes a directory.

(1) *Description*. The **rmdir** utility will remove the directory entry specified by each directory name operand, which must refer to an empty directory. Directories will be processed in the order specified. If a directory and a subdirectory of that directory are specified in a single invocation of **rmdir**, the subdirectory must be specified before the parent directory so that the parent directory will be empty when **rmdir** tries to remove it.

(2) *Format*. # **rmdir** [-options] dirname... [file...]

(3) *Example*. # **rmdir** /ajt/usr/ajt01/mydir
Removes the directory "mydir".

(4) *Options*.

-p Allow users to remove the directory and its parent directories which become empty. A message is printed on the standard error about whether the whole path is removed or part of the path remains for some reason.

-s Suppress the message printed on the standard error when -p is in effect.

6.3 File Management Commands.

6.3.1 Viewing File Contents. Solaris provides four commands for displaying and printing the contents of a file or files; **pg**, **more**, **pr**, and **cat**.

a. **pg** - Displays the contents of the specified file(s) in pages.

(1) *Description*. The **pg** command is a filter that allows the examination of filenames one screen full at a time. If the user presses the <Enter> key, another page is displayed. This command is different from other paginators in that it allows you to back up and review something that has already passed. The response that may be typed when **pg** pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment. Use the command **man pg** to determine the response for each category.

(2) *Format*. # **pg** [-options] [file...]

(3) *Example*. # **pg** /ajt/usr/ajt01/file1
Displays the contents of file1 a page full at a time.

(4) *Options.*

-number An integer specifying the size (in lines) of the window that pg is to use instead of the default. (On a terminal containing 24 lines, the default window size is 23 lines).

-p string Use the string as the prompt. If the prompt string contains a %d, the first occurrence of %d in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is the colon ":".

-e Does not pause at the end of each file.

-f Normally, pg splits lines longer than the screen width, but some sequences of characters in the text being displayed, such as escape sequences for underlining, generate undesirable results. The -f option inhibits pg from splitting lines.

+linenumber Start up at linenumber.

+/pattern/ Start up at the first line containing the regular expression pattern.

b. **more** - Browse or page through a text file.

(1) *Description*. The **more** utility is a filter that displays the contents of a text file on the terminal, one screenfull at a time. It normally pauses after each screenfull. Either --*More*-- or *file* is printed at the bottom of the screen. If **more** is reading from a file rather than a pipe, the percentage of characters displayed so far is also shown. The **more** utility scrolls up to display one more line in response to <Enter> being pressed. It

displays another screenfull in response to the *SPACE* bar being pressed. The **page** utility clears the screen before displaying the next screenfull of text; it only provides a one-line overlap between screens.

(2) *Format*. # **more** [-options] [file...]

(3) *Examples*.

more -c /etc/hosts
Clears the screen and displays the contents of the *hosts* file.
ls -ltr | more
Displays a long list, sorted by time stamp, in reverse order and piped to the **more** command.

(4) *Options.*

-c Clear before displaying. Redraws the screen instead of scrolling for faster displays. This option is ignored if the terminal does not have the ability to clear to the end of a line.
-d Display error messages rather than ringing the terminal bell if an unrecognized command is used. This is helpful for inexperienced users.
-s Squeeze. Replace multiple blank lines with a single blank line.
-f Do not fold long lines.
-l Do not treat FORMFEED characters <Ctrl-L> as page breaks.

c. **pr** - Prints files.

(1) *Description*. The **pr** utility is a printing and pagination filter. If multiple input files are specified, each is read, formatted, and written to standard output. By default, the input is separated into 66-line pages each with a 5-line header that includes the page number, date, time and the path name of the file and a 5-line trailer consisting of blank lines.

(2) *Format*. # **pr** [-options] [file...]

(3) *Examples*.

pr -d /etc/hosts | **lp**
Prints a double-spaced hosts file piped to **lp** (line printer) with a header.
ls -a | pr -n -h "Files in $(pwd)." | lp

Prints a numbered list of all files in the current directory piped the **lp** command
. # **pr -3d -h "File List"** file1 file2 | **lp**
Prints file1 and file2 as a double-spaced, 3-column listing headed by "File List" piped to **lp**.

(4) *Options*.

+page Begin output at the page number specified of the formatted input.
-d Produce output that is double-spaced. Append an extra NEWLINE character following every NEWLINE character found in the input.
-n[char][width] Provide width-digit line numbering. The default width is five (5).
-h header Use the string header to replace the contents of the file operand in the page header.

d. **cat** - Concatenate and display a file's contents.

(1) *Description*. The **cat** utility reads each file in sequence and writes it to the standard output.

(2) *Format*. # **cat** [-options] [file...]

(3) *Examples*.

cat file1
Prints file1 to the screen.
cat file1 file2 > file3
Prints the results of file1 and file2 concatenation to file3.

(4) *Options*.

-n Precede each line output with its line number.
-b Number the lines, as -n, but omit the line numbers from blank lines.
-u The output is not buffered. The default is buffered output.
-s cat is silent about non-existent files.

6.3.2 Solaris Utilities.

a. **wc** - Display a count of lines, words and characters in a file.

(1) *Description*. The **wc** utility reads one or more input files and, by default, writes the number of *NEWLINE* characters, words and bytes contained in each input file to the standard output. The utility also writes a total count for all named files, if more than one input file is specified. **wc** considers a word to be a non-zero-length string of characters delimited by white space. If no option is specified the default is **-lwc** (count lines, words and bytes).

(2) *Format*. # **wc** [-options] [file...]

(3) *Examples*.

wc /etc/hosts
Prints a count of lines, words and bytes for the *hosts* file to the screen.
wc -l file1 file2
Prints a count of lines for file1 and file2 to the screen.

(4) *Options*.

-l Count lines.
-w Count words delimited by white space characters or NEWLINE characters.
-c Count bytes.
-m Count characters.
-C Same as -m.

b. **tail** - Display the last part of a file.

(1) *Description*. The **tail** utility copies the named file to the standard output beginning at a designated place. If no file is named the standard input is used. Copying begins at a point in the file indicated by the option listed. The default is ten (10) lines.

(2) *Format*. # **tail** [-options] [file...]

(3) *Examples*.

tail /ajt/usr/ajt01/file1
Displays the last 10-lines of file1 (the default).
tail -20 /ajt/usr/ajt01/file1
Displays the last 20-lines of file1.

(4) *Options.*

-b Units of blocks.
-c Units of bytes.
-f Follow. If the input-file is not a pipe, the program will not terminate after the line of the input-file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input-file. Thus it may be used to monitor the growth of a file that is being written by some other process.
-l Units of lines.
-r Reverse. Copy lines from the specified starting point in the file in reverse order.

c. **grep** - Search a file for a pattern.

(1) *Description.* The **grep** utility searches files for a pattern and prints all lines that contain that pattern. Be careful using the characters $, *, [, ^, |, (,), and \ in the pattern because they are also meaningful to the shell. It is safest to enclose the entire pattern in single quotes (‘pattern‘). If no files are specified, grep assumes standard input. Normally, each line found is copied to standard output. The file name is printed before each line found if there is more than one input file.

(2) *Format*. # **grep** [-options] 'pattern list' [file...]

(3) *Example*. # **grep -in** 'ajt' /etc/hosts
Finds all occurrences of the pattern 'ajt' in the *hosts* file and prints them to the screen, ignoring upper and lower case distinctions and preceding each line by its line number in the file.

(4) *Options*.

-c Print only a count of the lines that contain the pattern.
-i Ignore upper and lower case distinction during comparisons.
-l Print only the names of files with matching lines, separated by NEWLINE characters. Does not repeat the names of files when the pattern is found more than once.
-n Precede each line by its line number in the file (first line is 1).
-v Print all lines except those that contain the pattern.
-h Prevent the name of the file containing the matching line from being appended to that line. Used when searching multiple files.
-s Suppress error messages about nonexistent or unreadable files.
-w Search for the expression as a word as if surrounded by \< and \>.

d. **find** - Find files.

(1) *Description.* The **find** utility recursively descends the directory hierarchy for each path seeking files that match a *Boolean* expression written in the primaries given below.

(2) *Format*. # **find** path [-options] string **-print**

(3) *Examples*.

find /usr -name "person*" **-print**
Search the /usr directory and find all files named *person** and print the results to the screen.
find / -name "daily*" **-print**
Begin at root and search all directories for files named *daily** and print the results to the screen.

(4) *Options*.

-name Specify the name of the file(s) being searched for.
-type Find only the type of file being searched for (-d for directory, -f for file, etc.).
-atime <u>number</u> Accessed time.
-ctime <u>number</u> When permissions were changed.
-mtime <u>number</u> Modified time.
-print Print filename and path to the screen.

e. **diff** - Display line-by-line differences between pairs of text files.

(1) *Description*. The **diff** utility compares the contents of file1 and file2 and writes to standard output a list of changes necessary to convert file1 into file2. This list should be minimal. No output will be produced if the files are identical.

(2) *Format*. # **diff** [-options] file1 file2

(3) *Example*. # **diff** file1 file2
Find the difference between file1 and file2

(4) *Options*.

-b Ignore trailing blanks (spaces and tabs) and treat other strings of blanks as equivalent.

-i Ignore the case of letters.

-t Expand TAB characters in output lines.

-w Ignore all blanks (SPACE and TAB characters) and treat all other strings of blanks as equivalent.

-c Produce a listing of differences with three lines of context.

6.4 File Access and Permission-Related Commands.

a. **chmod** - Change the permissions mode of a file.

(1) *Description*. Changes or assigns the mode of a file. The mode of a file specifies its permissions and other attributes. The mode may be absolute or symbolic.

(a) Besides the super-user (root), who is automatically granted access to all files, there are three categories of users associated with each file or directory (user, group and other). The user is the owner of the file. When a file or directory is created, Solaris keeps track of the User ID. Each user belongs to a particular group. A group might be several people in the same department or people working on a project together. Each group has a unique ID number and name. Other refers to any system user who is not the owner, nor a member of the owner's group. The symbols in the first column indicate the permissions. The permission symbols are "r" (read), "w" (write), "x" (execute), and "-" (no permission). By breaking down the column, we can see the permissions for each type of user. Figure 6-1 shows the permissions.

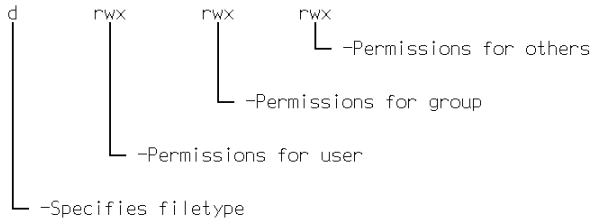

Figure 6-1. Permissions.

(b) Owners can change access permission to their files and directories. However, in granting such access, some of the system's inherent security is diminished. There are two notation types used to represent permissions:

1 *Symbolic Notation*. Single character symbols are used to represent permission coding as illustrated in figure 6-2.

USER CLASS	PERMISSION LEVELS	FUNCTION SYMBOLS
u = user (owner)	r = read	+ = add to
g = group	w = write	- = remove
o = other	x = execute	= = assigns

Figure 6-2. Symbolic Permissions Notation.

2 *Octal Notation*.

Numeric values are assigned to each permission level as follows:
read = 4 write = 2 execute = 1

These values are combined (added) to form the proper permission level as shown in figure 6-3.

rwx = 7	-wx = 3
rw- = 6	-w- = 2
r-x = 5	--x = 1
r-- = 4	--- = 0

Figure 6-3. Permission Levels.

Read (r) Permission - Controls who can read or copy the contents of a file, or look at the contents of a directory.

Write (w) Permission - Controls who can change the contents of a file by editing it or by redirecting output to it, or change the contents of a directory by adding, deleting, or changing file.

Execute (x) Permission - Controls who can execute a command file or who may change to or pass through a directory.

(2) *Format.* # **chmod** [-options] <mode> file...

(3) *Examples.*

chmod 777 file0

Give full permissions (read, write and execute) to the user, group and others for file0.

chmod a-x file1

Deny execute permissions to everyone for file1.

chmod 444 file2

Allow only read permissions to everyone for file2.

chmod go+rw file3

or

chmod 066 file3

Make a file readable and writeable by the group and others.

(4) *Options.*

-f Force. chmod will not complain if it fails to change the mode of a file.

-R Recursively descend through directory arguments, setting the mode for each file as described above. When symbolic links are encountered, the mode of the target file is changed, but no recursion takes place.

b. **chown** - Change file ownership

.

(1) *Description*. This utility will set the user ID of the file named by each file to the user ID specified by owner, and optionally, will set the group ID to that specified by group. If chown is invoked by other than the super-user, the set-user-ID bit is cleared. Only the owner of a file (or the super-user) may change the owner of that file.

(2) *Format*. # **chown** [-options] owner file...

(3) *Example.* # **chown ajt** file1
Changes the owner of file1 to "ajt"

(4) *Options.*

-f Do not report errors.

-h If the file is a symbolic link, change the owner of the symbolic link. Without this option, the owner of the file referenced by the symbolic link is changed.

-R Recursive. Descend through the directory, and any subdirectories, setting the ownership ID. When a symbolic link is encountered, the owner of the target file is changed (unless the -h option is specified), but no recursion takes place.

c. **chgrp** - Change file group ownership

(1) *Description*. This utility will set the group ID of the file named by each file operand to the group ID specified by the group operand.

(2) *Format*. # **chgrp** [-options] group file

(3) *Example*. # **chgrp sarss** file1
Changes the group for file1 to "sarss".

(4) *Options*.

-f Force. Do not report errors.
-h If the file is a symbolic link, change the group of the symbolic link. Without this option, the group of the file referenced by the symbolic link is changed.
-R Recursive. Descend through the directory and any subdirectories, setting the specified group ID. When a symbolic link is encountered, the group of the target file is changed (unless the -h option is specified), but no recursion takes place.

6.5 Disk Space Utilization Commands.

6.5.1 Disk Space and Usage.

a. **df** - Report the number of free disk blocks and files.

(1) *Description*. The **df** command displays the amount of disk space occupied by mounted or unmounted file systems, directories, or mounted resources, the amount of used and available space, and how much of the file system's total capacity has been used.

(2) *Format*. # **df** [-options] [directory | block-device | resource ...]

(3) *Examples.*

df
Reports on the file system.
df -b
Prints the available free kilobytes for each file system.

(4) *Options.*

-a Report on all file systems including ones who have the ignore option set.
-b Print the total number of kilobytes free.

-e Print only the number of files free.
-k Print the allocation in kilobytes.
-l Report on local file systems only.

b. **du** - Summarize disk usage.

(1) *Description*. The **du** utility writes to standard output the size of the file space allocated to each subdirectory of the file hierarchy rooted in each of the specified files. The size of the file space allocated to a file of type directory is defined as the sum total of space allocated to all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself. Files with multiple links will be counted and written for only one entry. The directory entry that is selected in the report is unspecified. By default, files sizes are written in 512-byte units, rounded up to the next 512-byte unit.

(2) *Format*. # **du** [-options] [file...]

(3) *Examples.*

du
Reports a summary of disk usage for all.
du -d /
Reports usage only on the root partition.

(4) *Options.*

-a In addition to the default output, report the size of each file not of type directory in the file hierarchy rooted in the specified file.
-k Write the file sizes in units of 1024 bytes, rather than the default 512-byte units.
-s Instead of the default output, report only the total sum for each of the specified files.
-d Do not cross file system boundaries. For example, du -d / reports usage only on the root partition.
-L Process symbolic links by using the file or directory which the symbolic link references, rather than the link itself.
-o Do not add child directory usage to a parent's total.
-r Generate messages about directories that cannot be read and files that cannot be opened, rather than being silent (the default).

6.6 Process Commands.

a. **ps** - Report process status.

(1) *Description.* The **ps** command prints information about active processes. Without options, **ps** prints information about processes associated with the controlling terminal. The output contains only the process ID, terminal identifier, cumulative execution time, and the command name. Otherwise, the information that is displayed is controlled by the options. Some options accept lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or else enclosed in quotes and separated by commas or spaces. Things can change while **ps** is running; the snap-shot it gives is true only for a split-second, and it may not be accurate by the time you see it. Some data printed for defunct processes are irrelevant.

(2) *Format.* # **ps** [-options] [arguments]

(3) *Example*.

ps -ef
Prints the UID, PID, PPID, C, STIME, TTY, TIME, and CMD of all processes now running.
ps -ef | more
Prints the same information piped to the more utility for all processes now running.
ps -ef | grep login
Prints the same information piped to the **grep** utility for all occurrences of "login". **ps -ef** or **ps -o stime** may not report the actual start of a tty login session, but rather an earlier time, when a getty was last respawned on the tty line.

(4) *Options*.

-a List information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal.
-A List information for all processes. Identical to -e.
-c Print information in a format that reflects scheduler properties.
-d List information about all processes except session leaders.
-e List information about every process now running.
-f Generate a full listing
-j Print session ID and process group ID.
-l Generate a long listing.

-y Under a long listing (-l), omit the obsolete F and ADDR columns and include an RSS column to report the resident set size of the process.

b. **kill** - Terminate a process.

(1) Description. The **kill** utility sends a signal to the process or processes specified by each *pid* operand. The signaled process must belong to the current user unless the user is the super-user.

(2) *Format*. # **kill** [-option] signal pid ...

(3) *Examples*.

kill -9 100 -165
kill -s 100 -165
kill -s KILL 100 -165

All three examples sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose process group ID is 165, assuming the sending process has permission to send that signal to the specified processes, and that they exist.

(4) *Options*.

-l Write all values of signal supported by the implementation, if no operand is given.
-s <u>signal</u> Specify the signal to send.
-9 Kill the process unconditionally.

<u>6.7 User Identification-Related Commands</u>.

a. **who** - Who is on the system.

(1) *Description*. This utility can list the user's name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current system user. The general output is the user's login name, the capability of writing to the terminal, the name of the line found in /dev, the time since the user login, the time elapsed since the user's last activity, the user's process-ID, a comment line, and the exit status for dead processes.

(2) *Format*. # **who** [-options]

(3) *Examples.*

who
Displays the name, line and time without a heading.
who -H
Displays the name, line and time with a heading.
who -a
Displays the name, line, time, idle, pid and comments with a heading.
who am i
Displays your name only.

(4) *Options*.

-a Display the named file with -b, -d, -l, -p, -r, -t, -T, and -u options turned on.
-b Display the date and time of the last boot.
-d Display all processes that have expired and not been respawned by init. The exit field appears for dead processes and contains the termination and exit values of the dead process. This can be useful in determining why a process terminated.
-H Output column headings above the regular output.
-l List only those lines on which the system is waiting for someone to login. The name field is LOGIN in such cases. Other fields are the same as for user entries except that the state field does not exist.
-m Output only information about the current terminal.
-n x Take a numeric argument, x, which specifies the number of users to display per line. x must be at least 1. The -n option may only be used with -q.
-p List any other process which is currently active and has been previously spawned by init. The state, line and idle fields have no meaning.
-q Quick who. Display only the names and the number of users currently logged on. When this option issued, all other options are ignored.
-r Indicate the current run-level of the init process.
-s Default. List only the name, line and time fields.
-T Same as the -s option, except that the state field is also written.
-t Indicate the last change to the system clock by root.
-u List only those users who are currently logged in.

b. **whodo** - Who is doing what.

(1) *Description*. Produces formatted and dated output from information in various files. The display is headed by the date, time, and machine name, user-ID and login time, followed by a list of active processes associated with the user-ID. The list

includes the device name, process-ID, CPU minutes and seconds used, and process name. If user is specified, output is restricted to all sessions pertaining to that user.

(2) *Format*. # **whodo** [-options] [user]

(3) Examples.

whodo
Displays user information.
whodo -l
Displays a long listing of user information.

(4) Options.

-h Suppress the heading.
-l Produce a long form of output. Fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in hours:minutes), the idle time (the time since the user last typed anything) (in hours:minutes), the CPU time used by all processes and their children on that terminal (in minutes:seconds), the CPU time used by the currently active processes (in minutes:seconds) and the name and arguments of the current process.

6.8 Print Commands.

a. **lp -** Submit a print request.

(1) *Description*. This command submits print requests to a destination. There are two formats of the **lp** command. The first form prints files and associated information, collectively called a print request. The second form changes print request options. This form of **lp** can only be used on a Solaris LP print server.

(2) *Format.* # **lp** [-options] [file...]

(3) *Examples.*

lp /etc/hosts
Sends a print request to the line printer assigned for 1 copy of the *hosts* file.
lp -n 10 /etc/hosts
Sends a print request to the line printer for 10 copies of the *hosts* file.

(4) *Options.*

-c Copy file before printing.
-d destination Print file on a specific destination.
-f form-name Print file on form-name.
-H special-handling Print the request according to the value of special-handling (hold, resume, or immediate).
-n number Print a specific number of copies of a file.
-o option Specify printer-dependent options (nobanner, nofilebreak, length, width, lpi, cpi, or stty).
-P page-list Print the pages specified in page-list in ascending order.
-p Enable notification on completion of the print request. Delivery of the notification is dependent on additional software.
-q priority-level Assign the print request a priority in the print queue.
-s Suppress the display of messages sent from lp.
-t title Print a title on the banner page of the output.
-w Write a message on the user's terminal after the files are printed.

b. **cancel** - Cancel a print request.

(1) *Description*. This utility cancels print requests. There are two formats for the **cancel** command. The first form of **cancel** has two optional arguments: print requests and destinations. The second form of **cancel** cancels a user's print request on specific destinations.

(2) *Format.* # **cancel** [-option] [request-ID...][destination...]

(3) *Examples.*

cancel ajt01-17
Cancels a print request for request-ID ajt01-17.
cancel -u ajt01!root
Cancels a specific user's (ajt01!root) print request.

(4) *Option.*

-u user The name of the user for which print requests are to be canceled. Specify user as a user-name.

c. **lpstat** - Print information about the status of the print service.

(1) *Description*. This command displays information about the current status of the LP print service to standard output. If no options are given, **lpstat** prints the status of all the user's print requests made by **lp**. Any arguments that are not options are assumed to be request-IDs as returned by **lp**. The **lpstat** command prints the status of such requests. Options may appear in any order and may be repeated and intermixed with other arguments. Some key letters may be followed by an optional list that can be in one of two forms: a list of items separated from one another by a comma, or a list of items separated from one another by spaces enclosed in quotes.

(2) *Format.* # **lpstat** [-options] [argument...]

(3) *Examples.*

lpstat -u "user1 user2 user3"
Prints the status on user1, user2 and user3.
lpstat -o all
or
lpstat -o
Prints the status of all output requests.

(4) *Options.*

-d Print the default destination for output requests.
-o [list] Print the status of output requests. list is a list of intermixed printer names, class names, and request-IDs.
-r Print the status of the LP request scheduler.
-R Print a number showing the position of each request in the print queue.
-s Print a status summary, including the status of the LP scheduler, the default destination, a list of printers and their associated services, a list of the machines sharing print services, a list of all forms currently mounted, and a list of all recognized character sets and print wheels.
-s Print all status information.
-u [login-ID-list] Print the status of output requests for users.
-v [list] Print the names of printers and the path names of the devices associated with them or remote system names for network printers.

d. **lpsched**, **lpshut** - Stop and start the LP print service.

(1) *Description*. The **lpsched** command starts or restarts the LP print service. The **lpshut** command stops the LP print service. Printers that are printing when **lpshut** is invoked stop printing. Printers that are restarted using **lpsched** reprint, in their entirety, print requests that were stopped by **lpshut**.

(2) *Formats.*

lpsched
lpshut

(3) *Examples.*

lpsched
Starts or restarts the line printer service.
lpshut
Stops the line printer service.

(4) *Options.* None

e. **lpc -** Line printer control program.

(1) *Description*. This program controls the operation of printers. Use **lpc** to start or stop a printer, disable or enable a printer's spooling queue, rearrange the order of jobs in a print queue, or daemon, and display the status of a print queue and printer. **lpc** can be run from the command line or interactively. All users may execute the following **lpc** commands:

?	abort	enable	disable	help	restart	status	topq	
clean	exit		down	quit	quit	start	stop	up

(2) *Format.* # **lpc** [command] [parameter...]

(3) *Examples.*

lpc ?
Displays all **lpc** commands.
lpc ? enable
Displays information about the **lpc** enable command.
lpc up ajt01
Enables queuing and printing on the printer assigned to ajt01.

(4) *Options.* None

f. **lpq** - Display the content of a print queue.

(1) *Description*. The **lpq** utility displays the information about the contents of a print queue. A print queue is comprised of print requests that are waiting in the process of being printed. It displays the user name, the position of a print request in the queue, the name of the file or files comprising a print request, the job number, and the size of the file in bytes. Normally, only as much information as will fit on one line is displayed. If the name of the input file associated with a print request is not available, the input file field indicates the standard input.

(2) *Format.* # **lpq** [-options] [request-ID] [user]

(3) *Examples.*

lpq
Displays print queue status for all jobs queued.
lpq -l ajt01-18
Displays information about job number ajt18.

(4) *Options.*

-P <u>destination</u> Displays information about printer or class of printers.
-l Displays information in long format.

g. **lpr** - Submit print requests..

(1) *Description*. The **lpr** utility submits print requests to a destination. It prints files and associated information, collectively called a print request. If the file is not specified, **lpr** assumes the standard input.

(2) *Format.* # **lpr** [-options] [file...]

(3) *Examples.*

lpr /etc/hosts
Prints the *hosts* file.
lpr -# 2 /etc/hosts
Prints two copies of the *hosts* file.

(4) *Options.*

-P destination Print the file on a specific printer or class of printers.
-# number Print a specific number of copies.
-C class Print class as the job classification on the banner page of the output. Enclose class in double quotes if it contains blanks. If class is not specified, the name of the systems is printed as the job classification.
-J job Print the job name on the banner page of the output. Enclose job in double quotes if it contains blanks. If job is not specified, the file name is printed as the job name on the banner page of the output.
-T title Print a title on the banner page. Enclose in double quotes if it contains blanks. If title is not specified, the file name is printed on the banner page.
-i number Indent the number of SPACE characters.
-1 |-2|-3|-4| font Mount the specified font in the font position 1, 2, 3, or 04. Specify font as a valid font name.
-w number Print file with pages of a specific width. number indicates the number of columns of width.
-h Suppress printing of the banner page of the output.
-s Use full pathnames for file name rather than trying to copy them.

h. **lprm** - Remove print requests from the print queue.

(1) *Description*. The **lprm** utility removes print requests from the print queue. Without arguments, it deletes the current print request. It reports the name of the file associated with print requests that it removes. It is silent if there are no applicable print requests to remove. Users can only remove print requests associated with their name. If a super-user executes **lprm** and specifies the user operand, it removes all print requests belonging to the specified user.

(2) *Format.* # **lprm** [-option] [request-ID...] [user...]

(3) *Examples.*

lprm ajt01-18
Removes print job number ajt01-18.
lprm ajt01!root
Removes all print jobs for the user "root".

(4) *Option.*

-P destination The name of the printer or class of printers from which to remove print requests. If a user specifies this option, remove all print requests owned by that user. If a super-user specifies this option, remove all requests in the print queue.

6.9 vi Screen Editor.

a. **vi**, **view**, **vedit** - Screen-oriented (visual) display editor.

(1) *Description*. **vi** (visual) is a display-oriented text editor based on an underlying line editor **ex**. When using **vi** changes you make on the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file. The **view** invocation is the same as **vi** except that the read only flag is set. The **vedit** invocation is intended for beginners. It is the same as **vi** except that the report flag is set to 1, the showmode and novice flags are set, and magic is turned off. These defaults make it easier to learn how to use **vi**.

(2) *Formats.* # **vi** [-options] file...
view [-options] file...
vedit [-options] file...

(3) *Examples.*

vi /etc/hosts
Allows you to edit the *hosts* file.
vi -R /etc/hosts
The *hosts* file is opened in read only mode, preventing accidental over writing of the file.

(4) *Options.*

-R Read only mode.
-r filename Recover the version of the file name that was in the buffer when a crash occurs. Edit the file name after an editor or system crash.

b. Modes, Edit Options and Commands.

(1) *Screen Commands*.

<Ctrl>L - Clear, redraw window.
<Ctrl>B - Displays previous page.
<Ctrl>F - Displays next page.

(2) *Cursor Movement*.

h - Moves cursor one position to left.
l - Moves cursor one position to right.
k - Moves cursor one line up.
j - Moves cursor one line down.
G - Moves cursor to last line of file.
0 - Moves cursor to beginning of line.
$ - Moves cursor to end of line.

(3) *Canceling/Undoing Commands*.

<Esc> - Exit text entry mode.
u - Undo previous change.

(4) *Text Entry Commands*.

A - Append text at the end of the line.
a - Append text after the cursor.
I - Insert text at the beginning of the line.
i - Insert text before the cursor.
O - Open a line above the cursor, and insert text.
o - Open a line below the cursor, and insert text.
s - Substitute characters.
S - Substitute lines.
R - Enters replace character mode.

(5) *Search Commands*.

/string - Search forward for matching string.
/ - Continue a previous forward search.
?string - Search backward for matching string.
? - Continue a previous backward search.
n - Continue a previous search in the same direction.
N - Continue a previous search in the opposite direction.

(6) *Commands for Text Modifications*.

x - Deletes a character at the cursor position.
J - Joins two lines.
r - Replaces a character at the cursor.
dd - Deletes the current line.
cw - Change a word.
y - Yanks the current line (into temporary buffer).
p - Puts the line one line below the cursor.
. - Repeats the last command.

(7) *Command Line (:)*.

:w - Writes the buffer contents to the disk.
:w! - Over writes the file with the contents of the buffer.
:q - Quits the editor.
:wq - Writes the buffer contents to the disk and quits the editor.
:wq! - Write, quit and save.
:set showmode - Sets the show mode feature.
:set number - Turns on line numbers.
:set nonumber - Turns off line numbers.
:set list - Shows a dollar sign to represent the end of the line.
:r filename - Reads a designated file into the current file.
<Shift>zz - If the file is modified, write and exit; otherwise, exit.

6.10 Solaris Communications Commands.

6.10.1 Messages.

a. **write** - Write to another user.

(1) *Description*. The **write** utility reads lines from the user's standard input and writes them to the terminal of another user. When first invoked, it writes the message "Message from sender-login-ID (sending-terminal) [date] ..." to the user. When it has successfully completed the connection, the sender's terminal will be alerted twice to indicate that what the sender is typing is being written to the recipient's terminal. If the recipient wants to reply, this can be accomplished by invoking the **write** command upon receipt of the initial message. Whenever a line of input as delimited by a NEWLINE, EOF or EOL a special character is accumulate, the accumulated data will be written on the other user's terminal. Permission to be a recipient of a **write** message can be denied

or granted by use of the **mesg** utility. The following protocol is suggested for using **write**: when you first write to another user, wait for them to write back before starting to send; each person should end a reply with "o" for over so the other person knows when to reply; each person should end the session with "oo" for over and out when the conversation is to be terminated.

(2) *Format.* # **write** user [terminal]

(3) *Example.* # **write** ajt02
Write to ajt02.

(4) *Options.* None

b. **wall** - Write to all users.

(1) *Description*. **wall** reads its standard input until an EOL and then sends the message "Broadcast Message from ..." to all currently logged-in users. If a file name is given, then the message is read in from that file. Normally, pseudo-terminals that do not correspond to rlogin sessions are ignored. Thus, when using a window system, the message appears only on the console window. It is used to warn all users, typically prior to shutting down the system. The sender must be a super-user to over ride any protections the users may have invoked.

(2) *Format.* # **wall** [-option] [file...]

(3) *Examples.*

wall /etc/motd
Writes the contents of the *motd* (message of the day) file to each logged on user.

wall -g sarss /etc/motd
Writes the contents of the *motd* file to each sarss group user that is logged on.

(4) *Options.*

-a Broadcast the message to the console and pseudo-terminals.
-g <u>group-name</u> Broadcast to a specified group only.

6.10.2 Electronic Mail.

a. **mail** - Read mail or send mail to users.

(1) *Description*. **mail** reads from the standard input up to an EOF <Ctrl-d> or, if reading from a terminal device, until it reads a line consisting of just a period and adds the letter to the mail file for each recipient.

(2) *Format.* # **mail** recipient

(3) *Examples.*

mail
Read mail addressed to you.
mail ajt02
Send mail to ajt02.

(4) *Options.*

-m message-type A Message Type: line is added to the message header with the value of message-type.
-t A To: line is added to the message header for each of the intended recipients.
-w A letter is sent to a remote recipient without waiting for the completion of the remote transfer program.

b. Reading **mail** user commands.

(1) While in the **mail** utility, use the <Shift>? to gain access to the user commands.

(2) The following commands are available.

Display message number.
- Print previous.
+ Next (no delete).
<CR> Next (no delete).
n Next (no delete).
! command Execute command.
a Position at and read newly arrived mail.

d [#] Delete message # (default is the current message).
dp Delete the current message and print the next message.
dq Delete the current message and exit.
h Display header information.
h a Display all headers.
h d Display headers of letters scheduled for deletion.
h [#] Display headers around # (default is the current message).
m user Mail (and delete) the current message to user.
p Print and over ride any warnings of binary content.
P Over ride default mode and display ALL header lines (default is brief).
q or <Ctrl>d Quit.
r [arguments] Reply to and delete the current letter via mail [arguments].
s [files] Save and delete the current message (default is the mbox).
u [#] Undelete message # (default is the current message).
w [files] Save and delete the current message without a header.
x Exit without changing mail.
y [files] Save and delete the current message (default is mbox).

6.11 Solaris-hosted DOS File/Utility Commands.

6.11.1 DOS Text File Conversion Command.

a. **dos2unix -** Convert text file from DOS format to ISO format.

(1) *Description*. Converts characters in the DOS extended character set to the corresponding ISO standard characters. This command can be invoked from either DOS or Solaris; however, the file names must conform to the conventions of the environment in which the command is invoked. If the original file and the converted file are the same, dos2unix will rewrite the original file after converting it

(2) *Format.* # **dos2unix** [-options] originalfile convertedfile

(3) *Examples.*

dos2unix dosheader1 unixheader1
Converts the dosheader1 file to the unixheader1 file.
dos2unix -ascii dosheader1 unixheader1
Removes extra carriage returns and converts the dosheader1 file to the unixheader1 file.

(4) *Options.*

-ascii Removes extra carriage returns and converts EOF characters in DOS format text files to conform to Solaris requirements.

-iso This is the default. It converts characters in the DOS extended character set to corresponding ISO standard characters.

-7 Convert 8-bit DOS graphics characters to 7-bit space characters so that Solaris can read the file.

b. Messages returned.

(1) "File *filename* not found, or no read permissions" is displayed when the input file you specified does not exist, or you do not have read permission. Check read permissions with the **ls -l** command.

(2) "Bad output file name *filename*, or no write permission" is displayed when the output file you specified is either invalid, or you do not have write permission for that file or the directory that contains it. Check also that the drive or diskette is not write-protected.

(3) "Error while writing to temporary file" is displayed when an error occurred while converting your file, possibly because there is not enough space on the current drive. Check the amount of space on the current drive using the **DIR** command. Also be certain that the default diskette or drive is not write protected. When this error is displayed the original file remains intact.

(4) "Translated temporary file name = *filename*" is displayed because the program could not perform the final step in converting your file. Your converted file is stored under the name indicated on the second line of the message.

(5) "Could not rename temporary file to *filename*" is displayed when true.

6.12 Help Command.

a. **man** - Find and display reference manual pages.

(1) *Description.* The **man** command displays information from the reference manuals. It displays complete manual pages that you select by name, or one-line summaries selected either by keyword (1k), or by the name of an associated file (-f). If no manual page is located, **man** prints an error message.

(2) *Format.* # **man** [-options] name ...

(3) *Example.*

man pwd
Prints help type information about the command pwd.
man pwd | **lp**
Prints help type information about the command pwd and pipes it to the line printer (lp).

(4) *Options.*

-a Show all manual pages matching name within the MANPATH search path. Manual pages are displayed in the order found.
-d Debug. Displays what a section-specifier evaluates to, method used for searching, and paths searched by man.
-f Attempt to locate manual pages related to any of the given files.
-k keyword ... Print out one-line summaries from the table of contents that contain any of the given keywords.
-l List all manual pages found matching name within the search path.
-M path Specify an alternative search path for manual pages.
-r Reformat the manual page, but do not display it.
-s section Specify sections of the manual for man to search.

b. **help** - Ask for help regarding the Source Code Control System (SCCS) error and warning messages.

(1) *Description.* **help** retrieves information to further explain errors messages and warnings from SCCS commands. It also provides some information about SCCS command usage. If no arguments are given, **help** prompts for one. An argument may be a message number, which normally appears in parentheses following each SCCS error or warning message, or an SCCS command name. help responds with an explanation of the message or a usage line for the command. When all else fails, try " /usr/ccs/bin/help stuck | more".

(2) *Format.* # **help** [msg-code]

(3) *Example.*

help msg-code
Prints help type information about the message code.

(4) *Options.* None

6.13 MSDOS/Solaris Command Cross Reference. This paragraph provides a summary of selected Solaris commands and their DOS equivalents.

Functional Description	**MS/DOS**	**Solaris**
List directory contents	dir	ls -la
List current directory	prompt #p	pwd
Create new directory	md	mkdir
Remove empty directory	rd	rmdir
Change default directory	cd \	cd /
Display file contents	type	cat, pg, and more
Concatenate files	copy file1 + file2 file3	cat file1 file2 > file3
Append files	type file1 >> file2	cat file1 >> file2
Check free disk space	chkdsk	du
Print a file	print	lp
Copy	copy	cp
Compare files	comp	cmp
Echo message to screen	echo message	echo message
Erase a file	del	rm
Check the date	date	date
Check the time	time	date
Rename a file	ren	mv
Search file for pattern	find /N "pattern" file	grep "pattern" file
Count number of file lines	find /C "pattern" file	wc -1