

## SA135-285 - 6.14.99

This file was created several years ago from SUNs' SA-135 and SA285 courses (which are now called something else). These courses are the core of SUNs' SA training curriculum. The courses are in two 8.5x11x1" notebooks which are difficult to lug around, have a lot of white space, and are not in electronic format to search, etc. The information is taken almost verbatim from the books, with a few paragraphs left out that I felt were just too fundamental to transcribe. These are just personal notes, and weren't originally created to be published.

Please remember that the information contained in this file, which is nearly verbatim, is the sole property of SUN. So don't publish it, or they will come whining to me. Outraged at the millions they are losing from my thievery. You get the idea?

Most of the images from the course were scanned, and are referred to in this document, but were removed because they made the file too unmanageable. Please contact me, and I will get you copies of them.

Feedback, etc, are welcome.

Hope this helps you.

Scot Corrie

[corrie@colltech.com](mailto:corrie@colltech.com)

### [LEGEND]

---

**::ls** = command  
**[:/etc/hosts]** = file  
**[ ]** = subject or a description  
**( )** = supplemental information  
**=** = exact definition of a term  
**=>** = syntax of a command  
**>** = type this  
**Ex:** = example  
**xxxx** = sample text  
**===** = Placeholder for whatever  
***italic*** = switch, or type to key in  
**R]** = Return  
**S]** = Shift  
**C]** = Control  
**>>** = Screen output or a reply

---

**cd** cd /net/===(/net/river1/export/home/corries)  
[go to a network drive]  
**find, chown** find / -user 225 -exec chown root {} \;  
[find all files belonging to a user, chown to root]  
**find** find / -atime +365 -size +1000000c - print  
[find and print a list of all files older than 365 days, larger than a meg]  
**find, ls** find / -nouser -exec ls -la {} \;  
[find all files without a current user]

### **::SA 135::**

axxiii % = C Shell  
\$ = Bourne and Korn Shell  
# = Superuser  
Three main parts of the OS: Kernel, Shell, File system  
a1.3 daemon - Spawns child processes at specific times. Normally they sleep.  
Ex: the init.d daemon initializes Internet processes.  
ONC - Open Network Computing (Public Networking Protocols)  
a1.4 Kernel - Details (manages hardware)  
a1.5 Shell - Details (interface between user and the kernel)

- a1.6 **[File system structure]**  
 /  
 [root]  
**/dev**  
 [device directories and files, which provide access to devices such as disk (dsk) and tape (rmt)]  
 [Tape Drives]  
 ?/rmt  
 [Disk Drives]  
 ?/dsk  
**/export/home** - [local home directories]  
 .../loginid1  
**/etc** - [system administration files such as *passwd* and *hosts* files]  
 ?/default - [files that control system access]  
 ?/default/cron -  
 ?/default/login -  
 ?/default/su -  
 ?/init.d - [files that control daemons, establishes how the system behaves]  
 ?/skel - [files for customizing the environment]  
**/home** - [exported home directories]  
**/kernel (/genunix {Solaris})** - [system kernel]  
 ?genunix -  
**/opt** - [unbundled and third-party software]  
**/usr** - [3<sup>rd</sup> party, user commands and programs and executable commands, system administration utilities, library routines and OpenWindows]  
 ?/bin - [command line executables]  
 ?/openwindows/bin - [command]  
**/var {variable}** - [spooling files such as printer and mail. various]  
 ?/adm - [error logs, files that grow]
- a1.9 **[Terms]:**  
 host = a computer system that you can log into  
 host name = a unique name for a system, each system on the network must have its own host name  
 client = a host or process that uses the services from one or more servers on a network. Can be both.  
 Server = a host or process that provides resources to one or more clients on the network. Can be both.  
 Network = a group of computers that, in a Sun environment, are connected using Ethernet.  
 ip address = a number used by networking software to identify machines attached to the network.  
 Node = entity that can't be logged into. (printer)  
 [Note: man pages are often setup as NFS drives on one system.]
- a2.9 **::groupadd** > groupadd -g 18000 infotech  
**::useradd** > useradd -u 109 -g 18000 -d /export/home/corries -c "Scot Corrie" -m -s /bin/ksh corries
- a3.7 [command line syntax] = command *option/s argument argument*  
 command=exe  
*options*=modify the exe;  
*argument*=the file or directory(path).  
 (many commands do not require all three parts)
- a3.10 [path name abbreviations]  
 . = current working directory  
 .. = moves up one  
 ~ = absolute path to home directory (only in KO and C)
- a3.11 **[File Types]**  
 / = directory  
 \* = exe  
 @ = symbolic link  
 none = plain text file (ASCII)
- a3.14 **::ls** > ls *options pathname*  
 -a = hidden  
 -b = block special (dev)  
 -f = extensions  
 -F = display file types - (/dir, \*exe, @link)  
 -l = long  
 -ld = shows directory

-p = create multiple levels of directories at one time

-R = directory and all subdirectories

Ex: > ls -l /kernel

```
drwxr-xr-x      2      root   sys      2048   Oct 24 11:10  drv
(FT,permissions link count  owner  Group owner size last mod.  File name)
(FT = File Type)
```

**[file types]**

- = file

d = directory (dev)

b = block special (dev)

c = character special

l = symbolic link

p = named pipe, or stream, which is used for inter-process communication

a3.18 **[ls wildcards]** >

\* = 0 or more

**s\*** = star, street

**i\*** = replace names for directories

**?** = single character

**0?** = 0a,0f,0q

a5.4 **[list a range of files]** > ls [b-f]\*

a3.15 [three parts of a file] = directory entry, inode, datablocks

<u>directory entry</u>	<u>inode</u>	<u>Data blocks</u>
file name, #	linked	actual data
	permissions	
	owner	
	group	
	size	
	mod times	
	pointers to	
	data blocks	

a4.3 **::cat** = [show the contents of a file] > ls -l | cat -vte (verbose, show hidden & control characters)

a4.4 **[control characters]**

C]-s = stops screen output

C]-q = starts screen output

C]-c = interrupts current activity

C]-d = indicates end-of-file or exit

C]-u = erases the command line

C]-w = erases the last word on the line

C]-z = suspends a job

(when displayed on the screen, the Control key is represented by “^”).

[ignore eof] > set -0

a4.5 **::more** > more ===/s

[scrolling keys]

spacebar = to scroll to the next screen

Return = to scroll one line at a time

b = to move back one screen

f = to move forward one screen

h = display a help menu of more features

q = to quit, and return to the shell prompt

/string = to search forward for string

a4.6 **::man** > man -s# passwd

[#]

1 = user commands

2 = functions

3 = system files

4 = file formats

5 = misc.

6 = demo software

7=device files

8=im

```

9=glossary
> man more =
> man -k keyword
> man -f mv cp rm
= /usr/share/man/windex whatis (location)
> # catman -w & [reformat entire man pages & create a database]
a4.7 ::head > head -n ===/s [display the first n lines of a file. 10 is the default if -n isn't used.]
Ex: > head -5 /etc/hosts
a4.8 ::tail > tail +n ===/s [display the last n lines of a file. 10 is the default if -n isn't used.]
a4.9 ::touch > touch -c * .*
a4.10 ::cp > cp -options source destination
-l = interactive
-r = +sub directories (recursive)
[copy a directory & subs to a different volume]
> cp tar cvpf - fusion | ( cd /export/home/corries/-images; tar xvpbf- )
[copy multiple files into a directory] > cp /etc/motd /etc/system sysfiles
a4.11 cp - [copy a directory and all its files to a new directory name in the current directory]
> pwd
>> /export/home/scot
> ls -a /etc/skel
>> .profile local.cshrc local.login local.profile
> cp -r /etc/skel .
> ls -a skel
.profile local.cshrc local.login local.profile
[copy the contents of a directory using a wildcard]
> pwd
>> /export/home/scot
> mkdir default
> cp /etc/default/* default
> ls default
>> cron fs init login passwd su tar utmpd
a4.12 ::mv > mv -l(") - (to move or rename files)
[ rename a file]
> mv project donnas_upgrade
[ move several files into a directory]
> mv projection research ../admin
[move a directory]
> mv maildir admin
a5.2 metacharacters > ; $ > ~ * ? [] < |
a5.3 ; = separator, to run several commands at a time
a5.4 [ ] = an argument, !(negates) > ls -r .[!]*
a5.5 > >> < | = output, dual output, input, pipe
> redirect the output of a command to a file
>> append the output of a command to the end a file
< redirect the input of a file to a command
| take the output of one command and pass it as input into a following command
a5.6 [redirect the output of the list command to a file]
> ls /etc > === (etc.list - will overwrite an existing file)
[redirect the input of a file to a command]
> mailx ===(ssa2@saturn) < /etc/hosts
[ ? ]
> find / -name .profile
> find / -name .profile; 2 > /dev/null
> find / -name .profile > ~/hits 2>~/misses or errors
> find / -name .profile | tee hits 2>~/find errors
set -0 noclobber (won't erase data)
cat /etc/hosts > | ~/my-hosts |(override noclobber) ~/===(my_hosts)
a8.4 ::vi = There are three modes of operation in vi
command mode; entry mode; last-line mode (colon mode (ex) );
open a file = command mode

```

> i a o = entry mode (type in the changes)  
> <esc> = command mode  
> : / ? = last-line mode  
> R] = command mode

### **vi Map**

a7.28 draw map

### **vi basics**

add text        a i O o R  
delete text     c w r  
paste text      p P  
delete text     x dd dw  
misc.            :wq!  
                  :q!  
                  :e!  
                  :r==(file)  
                  /text n <esc>  
                  :! <UNIX command>

a8.5 **[open a file]** > vi ===  
[recover a crashed file (onetime only)] > vi -r ===  
[open a file as read-only] > view ===

a8.6 **[input commands]**  
a - append text after the cursor  
A - append text at line end  
i - insert text before the cursor  
I - insert text at the beginning of the line

### **[positioning commands]**

h - left one character  
j - down one line  
k - up one line  
l - right (forward) one character  
H - top of screen  
L - bottom of screen  
C]-f - page forward one screen  
C]-d - down one-half screen  
C]-b - back one screen  
C]-u - up one-half screen

a9.3 Initialization files contain a series of commands that are run when a shell is started. Customize the environment.

### a9.4 **[Shell features]**

The initialization files maintain shell features between shells and subshells. They also maintain shell features from login session to login session. There are two levels of initialization files, system and user. At login, the system initialization files are read, and then the user initialization files are read. The tasks defined in the initialization files are executed. The initialization files maintain two types of shell features:

Local(shell) and Environmental (shell).

Local shell features, such as custom commands known as alias, are only available in the shell in which they are created unless placed in an initialization file that is read every time a shell is opened. Environmental shell features, such as exported variables, are available in the shell in which they were created and to any subsequent child shell.

Environmental features are inherited from the parent shell and terminated upon exit of the parent shell.

Environmental features are placed in initialization files read upon setup of the default login shell.

Custom shell features are terminated upon logging out. Initialization files are required to maintain custom shell features from login to login. The user initialization files are stored in the home directory of the user.

a9.5 **[shell features]** - Controls or sets: the prompt, printer, permissions, history, terminal, mail, OW, noclobber, path, custom commands.

The initialization files provide great flexibility to the user for customizing the environment. Generally, these files are set up as templates by the system administrator, and the user modifies them.

### a9.6 **[login sequence]**

[art/sa135\\_09-06.gif](#)

a9.7 **[init files]** - The following files must be in the users home directory:

Bourne = /etc/profile, .profile

Korn = /etc/profile, .profile, .kshrc

<u>Shell</u>	<u>Read During Login</u>	<u>Read Opening a Shell After Login</u>
--------------	--------------------------	---

Bourne	/etc/profile and .profile	
Korn	/etc/profile, .profile, .kshrc	.kshrc
C	/etc/.login, .cshrc, .login	.cshrc

A variable, ENV, must be assigned and exported in .profile for the Korn Shell to read .kshrc (in the users home directory)

a10.9 > ENV=\$HOME/.kshrc; export ENV  
\$HOME - means relative to the user  
C-Shell = /etc/login, .login, .cshrc

a9.8 **[shells]** - the login shell determines which init files are read during login. The Bourne shell uses .profile; the Korn shell uses .kshrc. At login, the .profile file is read. If you assign and export a ENV variable, the .kshrc file is read. .profile is read only during login. .kshrc is read every time you login.

a10.3 **[variable]** - A placeholder for information to be used by the system or user (i.e. default printer, path, etc.). Two categories are local (shell) and global (environment). Local is good only for the current (source) shell.

[strip man command] = man ksh | col -b | expand > kshfile  
-b = removes hidden control characters and reverse video.

| expand = converts ^I into physical files  
to erase these files later > rm \*\_file

a10.4 **[set a variable]** > VARIABLE=VALUE; export VARIABLE  
> unset VARIABLE

The Bourne and Korn shells use capital letters for built-in shell variable names. The first command format set the variable based on a name and value selected by the user, while the unset command remove the variable from the current shell and subshells.

The dollar sign (\$) metacharacter preceding a variable name enables the system to use the value of the variable and not the name of the variable.

a10.5 **::set** - displays local and global shell variables

a10.6 **[exporting variables]** - Solaris software provides several predefined environment variables. Exporting variables makes the variables available to the current shell and all of its subshells.

**::LPDEST** > LPDEST=it; export LPDEST (test by typing echo \$LPDEST) > unset LPDEST  
[setup default printer]

[exrc not read] > EXINIT='set showmope'; export EXINIT (sets a last line option for vi)

a10.7 **::env** - shows system wide environmental variables, not the local settings.

a10.8 **::PATH** > PATH=/usr/bin:/usr/openwin/bin: /etc:.  
> export \$PATH

The dot (.) in the PATH variable enables the system to look in the current working directory for commands.

(.) - current user directory, make this the last item

[how to set the path variable]

PATH=\$PATH:/usr/ucb (these happen to be Berkley commands)

[to add to the existing path]

PATH=\$PATH:~/bin: (keeps path)

a10.10 **::ENV** - defines the path to the .kshrc file. This variable must be stored in the .profile file to inform the system that the .kshrc file exists and is to be read when creating a Korn shell.

> ENV=\$HOME/===; export ENV

[set the .kshrc variable]

> ENV=\$HOME/.kshrc; export ENV

This command sets the environment variable to print to the .kshrc file in the \$HOME directory. HOME is a variable that is defined by the system to be the absolute path to the user's login directory. Using the dollar sign (\$) metacharacter preceding the HOME variable enables the system to use the value of HOME in the specified location.

a11.3 **::HISTORY** - ~/.sh\_history is automatically put in root

**HISTFILE** > HISTFILE= \$HOME/.sh\_history; export HISTFILE, **HISTSIZ** = 450; export HISTSIZ

a11.4 > HISTORY -4 = shows the last four

a11.5 **::R** > R V, R CP, R 23 - custom command in the Korn shell which enables you to re-execute from the history list.

a11.7 > set [+ ] o vi ( in .kshrc)

a11.8 **::alias** > alias aliasname=value (Ex: alias h=HISTORY; alias ls = 'ls -aFr')

a11.10 [delete an alias] > unalias aliasname

a11.11 [quotation marks] - control how the shell interprets the command

[**\$**] - to expand a variable, place at the beginning of the variable. (Ex: > W=Hello, echo \$W (Hello))

[grave accents] - '=== ' command substitution. To identify a command to the shell.

[**"x"**] - means to use the text inside literally, but allow for variable expansion (\$), and command execution ("")

(Ex: > echo "\*\*\*\$W \$LOGNAME\*\*\* hostname is 'uname -n\*\*\*' => \*\*Hello JohnnyD\*\*\*hostname is mariner\*\*\*\*\*  
 ['x'] - means interpret metacharacters literally (take away all meaning).

a11.12 ::PS1 (prompt) > PS1="moveit\$ "  
 => moveit\$ \_  
 > PS1="uname -n' !(current event)\$ "  
 => fusion 41\$ ||  
 > PS1='\$PWD\$ '  
 => /export/home/corries \$ \_)

a12.3 **[initialization files]** - contain a series of commands that are executed when a shell is started. An initialization file customizes the environment for that shell. There are two types of initialization files: system and user. System initialization files are maintained by a system administrator and reside in the /etc directory. The system initialization file for the Bourne and Korn shells is /etc/profile. The system initialization file for the C shell is /etc/login. Templates for the user initialization files are kept in the /etc/skel file.

Shell	System (Read First)	User (Read Second)	Template
Bourne	/etc/profile	\$HOME/.profile	local.profile
Korn	/etc/profile	\$HOME/.profile	local.profile
		\$HOME/.kshrc	
C	/etc/.login	\$HOME/.cshrc	local.cshrc
		\$HOME/.login	local.login

a12.4 **[:/etc/.profile script]** - at user login, system reads /etc/.profile first, then it reads the user's .profile file. This means that the user's preferences for variable settings can overwrite the default settings that appear in the /etc/.profile file.

Sets default settings such as:  
 LOGNAME(login name),  
 PATH (default path),  
 TERM (terminal),  
 Shows /etc/motd,  
 Sets the default permissions  
 Checks the Mail

a12.5 **[:/etc/skel]** - contains templates of initialization files for BO and C users. (local.cshrc, local.login, local.profile) Admintool automatically copies the related shell "local" file or files to the users home directory, and renames the file/s to the corresponding hidden init file/s. Not for KO shell.

a12.6 **[:/etc/local.profile]** - It's a good idea to take the /etc entry out of the path statement. [good diagram]  
 Changes to make:

PATH = take out the /etc entry for the average user  
 EXINIT='set showmode' (no excrc) if for remote users

a12.7 [ . ./profile or source] - To make the changes you did in .profile active.  
 BO or KO > . ./profile; or > . ./kshrc);  
 C > source ~/.login; or > source ~/.cshrc)

a12.8 [BO,KO,C maps]

Feature	BO	C	KO
Aliases	N	Y	Y
Command-line editing	N	N	Y
History List	N	Y	Y
Ignore Control-d(ignore eof)	N	Y	Y
Separate initialization file form .profile	N	Y	Y
Job control (ability to move job to and from the background and foreground for processing; ability to suspend a job)	N	Y	Y
Logout file	N	Y	N
Protect files from overwriting(noclobber)	N	Y	Y

a13.3 ::find {magnifying glass} > find *path expression action*  
 -name *filename* **[find all files with a certain name]** > find ~ -name ..sh\_history -print  
 -mtime *num\_days*; [find a file exactly 24 hours ago] > find . -mtime 1 (+,-,>,< 24 hrs)  
 [find any file . > 10000c = good for hidden viri] > find /etc/ -size +10000c (-<,+>)  
 -user *username*; [find all files owned by a user] > find / -user corries -print  
 -atime *n* (access time); [find a file modified more than seven days ago] > find / -atime +7 -print  
 -ctime; [find the change time "itime really" of a file]  
 [finds the file and lists] > find ~ -name .profile -exec {same as -ls} ls -l {} \  
 -exec *command* {} \;

-ok *command* {} \; (interactive)  
-type f(file), d(dir), l(symbolic link), b (block spec), c(character), delete (find orphans)  
-nouser (files not belonging to a user in /etc/passwd),  
-nogroup (files not in /etc/group)

a13.5 **[find all files ending in tiff, start in /usr]** > find /usr -name '\*tif' -print  
[find and remove core files] > find / -name core -exec rm {} \; ; [^\core dump]  
[find from root, all files not changed in the last 90 days] > find /export/home -mtime +90 -print  
[find files larger than 400 blocks (@200k)] > find /export/home -size +400 -print  
[in case a users directory is deleted, but someone else had files in it use the following] :  
> find /export/home/=== -exec mv {} /tmp/=== \;

a13.6 **::grep** [microscope, Global Regular Expression Print] => grep *options string filename*  
(Use the grep command to search a file for a specified text string)  
[any file with June 10th date] > grep fission /etc/hosts=line/s ls -la |grep 'June 10'  
[regular expressions] > man -S 5 regexp  
^ = beginning of line  
\$ = end of line

. = any single character  
\* = 0 or more of previous character  
.\* = 0 or more of any character  
[ ] = [r|z9] (specific), [a-m], [^0-9] (inverts search. i.e. not 0-9)  
“^[^0-9].\*” = any line that starts with something not a 0.

a13.7 **::tar** - [kind of like a binder you put files into.] (Tar first, then compress).  
tar => tar *options [output file] filename(s) /directory(s)*  
c = new file  
t = t.o.c.

x = extract the specified files from the tar file  
f = name of file or drive  
v = verbose  
p = leave modification date  
Ex: cd /export/home, tar cvf /dev/rmt/0 corries  
[directories; ls and grep] > ls -l | grep “^d”  
[ ] > ls -F | grep “/\$”  
[find all dots] > ls -a1 | grep “^\.”  
[prepends line #] > grep -n  
[-1=name of file] > grep -l “PATH .”  
[How to save a directory to tape]  
> cd /export/home  
> tar cvf /dev/rmt/0 corries

a13.8 **::compress** > compress -v bin.file ( becomes bin.file.Z)  
**::uncompress** > uncompress bin.file.Z  
(you cannot compress a directory. Tar then compress.)

a13.9 **[backup a home directory]**  
> cd /export/home  
> tar cvf /tmp/home.tar scot  
[check t.o.c.] > tar tvf /tmp/home.tar  
[compress] > compress -v(shows to screen) /tmp/home.tar  
[tar to a tape] > cd /tmp  
> tar cvf /dev/rmt/0 home.tar.Z  
**::restore**  
> mkdir newhome  
> cd newhome

[extract the home directory from tape] > tar xvf /dev/rmt/0 home.tar.Z  
[uncompress the file] > uncompress home.tar.Z  
[extract the tar file] > tar xvf ./home.tar .kshrc or “.”

a14.6 **[::permissions]** - all files and directories have a UID with a single value. (> ls -n displays the UID and GID)  
Every system process also has a UID and GID When a process attempts to read, write, or execute a file, the process' system data information is compared to the files' or directories' UID, and then to the GID. If neither matches, then the *other* category of permissions is used.

**System process=File or directory**

UID=UID\_\_\_\_\_ Yes\_\_\_\_\_ Use user permissions

No

GID=GID\_\_\_\_\_ Yes\_\_\_\_\_ Use group permissions

No\_\_\_\_\_ Use other permissions

a14.11 **chmod** => `chmod mode filename`

<b>(mode)</b>	<b>who</b>	<b>-op</b>	<b>-permissions</b>
u	- owner	=	- set permissions
g	- group	-	- remove access
o	- other	+	- give access
a	- all		
			r - read
			w - write
			x - execute

[deny read permissions to others] > `chmod o-r corries`

[only root can see] > `chmod 600 /marketing`

[user/group can see] > `chmod 770 /marketing`

[add execute permission for owner, and read permission for group and others] > `chmod u+x, go+r fission`

a14.13 **Octal (Absolute Mode) Chart**

4 = Read, 2 = Write, 1 = Execute  
7/ rwx 6/ rw- 5/ r-x 4/ r-- 3/ -wx 2/ -w- 1/ --x 0/---

a15.3 **umask** = This entry is in .profile. Determines the default permissions for files and directories. The permissions are assigned during the creation of new files and directories. Each digit represents the permissions to be denied, by default, when creating files and directories.

A15.4 **umask File Directory**

022	Gives read and write permission to the owner of the new file and read permission to group and others.	Gives all permissions to the owner and read and execute permissions to group and others.
027	Gives read and write permission to the owner of a new file and gives read permission to group, and no Permission to others.	Gives all permissions to the owner read and execute permissions to group, and no permission to others.

a15.6 Note: umask 027 must be in .profile to be permanent

[change umask] >

`vi .profile`

`umask 027`

`././profile`

a15.7 **ACL** = Access Control Lists provide greater control over file systems. The traditional UNIX file protections provides read, write, and execute permissions for: owner, group, and other. An ACL enables you to define file permissions for the owner, owner's group, other, specific users and groups, and default permissions for each of those categories.

a15.8 **setfacl** => `setfacl options acl_entry === [=== . . .]`

-f = copy acl to hundreds of files

-m = creates an ACL,

-s = replaces the entire ACL with the new ACL

-d = deletes ACL entries

-acl\_entry = ACL entry, defined below

=== = file or directory on which to set ACL entries

Basic ACL Entries

**ACL Entry Meaning**

u [ser] ::perms The owner's permissions.

g [roup] ::perms Permission for the owner's group.

o [ther] :perms Permissions for users other than the owner or members of the owner's group.

m [ask] :perms The ACL mask. The mask entry indicates the maximum permissions allowed for users (other than the owner) and for groups. The mask is a quick way to change permissions on all the users and groups.

u [ser] :uid:perms Permissions for a specific user.

g [roup] :gid:perms Permissions for a specific group.

( u:larryd:perms )

- a15.9 [add r/w permissions for ===] > setfacl -m user:corries:6 chi.doc  
 [check to see if a file has an ACL] > ls -l chi.doc (+ sign to right of the mode field)  
 [delete an ACL entry] > setfacl -d user:corries:6 chi.doc
- a15.10 **::getfacl** - [verify an ACL was set on the file]  
 => getfacl *options* === [=== ...]  
 -a = (default) shows name, owner, group, and ACL entries  
 -d = only shows the name, owner, and group  
 Ex: > getfacl -F fromfile tofile
- a15.11 [Ex: set the user permissions to read/write, group permissions to read only, and other permissions to none on ===. In addition, the specific user corries is given read/write permissions on the file, and the ACL mask permission is set to read/write, which means no user or group can have execute permissions.]  
 Ex: > setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:corries:rw- ===  
 (ACL's don't have to have masks)
- a15.12 **::SetUID** and **::SetGID** - Must be compiled binary files only. The owner and SU can set on a file and setgid permissions on a directory. These modifications enable you to control the modifications of files and create shared directories.
- Executable Programs  
 If a program has setuid permission, anyone who has permission to run the program is treated as the owner.  
 If a program has setgid permission, anyone who has permission to run the program is treated as if they belong to the program's group.  
 Executable programs with setuid or setgid permission get their UIDs or GIDs from the owner and group of the program file, instead of inheriting their UIDs and GIDs from the process (usually a shell) that started them. This is used when a program must access files that are normally only accessible to the owner or group owner of the program. In this case, the executable programs defines the interaction with the file/s being modified.
- Directories  
 Directories that have setgid will transfer this to any file/directory created below it. This is good for shared project directories.
- a15.13 **::setuid** and **::setgid** = The setuid and setgid bits are displayed as the letter **s** in the execute field for owner and group.  
 Ex: > ls -l /bin/passwd /etc/passwd /etc/shadow  
 >> -r-sr-sr-x 1 root sys 22208 Mar 27 06:21 /bin/passwd  
 >> -r----- 1 root sys 529 Mar 26 09:57 /etc/shadow  
 These permissions enable users to change certain fields in the /etc/shadow files when using the passwd command. If a capital **S** appears, it is an error condition indication that the setuid or setgid bit is on and the execute bit is off.  
 The setuid and setgid permissions are set with the chmod command using either symbolic or numeric notation for files. Numeric notation requires four octal numbers when specifying setuid or setgid and uses the left-most number to refer to these special permissions.  
 > 4=setuid, 2=setgid, 1=sticky bit, S(squirly)=problem  
 Ex: > chmod 4660 filex (exe - compiled binary)
- a15.14 **[control file modification]** > chmod 4755 setuid\_program or > chmod 2755 setgid\_program  
 [create shared directories] - (must be symbolic, files can be octal also).  
 > chmod g+s some\_directory  
 - The setgid bit on a directory must be set or changed using symbolic notation.  
**::sticky bit** - If a directory is writeable and has sticky bit set, files in it can be removed or renamed only if one of the following or more is true:  
 The user owns the file, or directory, has write permissions to the file, or is SU. (Prevents users from deleting other users files from public directories such as /var/tmp. Don't need to set this on files.  
 (Note: Set sticky bit on tmp)
- a15.15 **[sticky bit]** > displayed as a "t" in the execute field for others. A capital "T" is an undefined bit state, which means the sticky bit is on and the execute is off. This is not good!  
 Ex: > ls -ld /var/tmp  
 # drwxrwxrwt 2 sys sys 512 may 26 11:02 /var/tmp  
 [set sticky bit - This is something you may only do five times in your career] > chmod 1777 project  
 > chmod a=rwxt project
- a16.4 UID (in /etc/passwd) - Provide authentication for login procedures. Identify ownership of files and directories. UIDs 0-99 are reserved for special system accounts. The reserved system accounts are used to provide exclusive file access for certain system services such as the lp print service daemon.  
 GIDs identify group membership of users, files, and directories. A user may belong to up to 16 groups-one primary, fifteen secondary. (login is in /etc/passwd). GIDs 0-99 are reserved for special accounts such as bin,

- others, and staff user accounts. The reserved GIDs are also used to provide exclusive group file access.
- a16.5 **::id** => id *options* (Use to identify your user ID, user name, group ID, and group name.)  
Ex: > id -a corries (-a command shows all groups *corries* is member of.)
- a16.8 **[keep track of who is logging in as Superuser]** > cat /var/adm/sulog
- a16.9 **[su - username]** - causes the system profile in /etc/.profile to be used, then the specified users profile (\$HOME/.profile) to be used.  
> /usr/bin/who am I = who I logged in as  
> /usr/ucb/whoami = who I am at the moment
- a16.10 **::chown** => chown *user\_name* === or chown *UID* ===  
[change ownership of a directory] > chown -R new\_user\_name:(new group) *directory\_name*  
> chown -R jones sandss
- a16.11 **::chgrp** => chgrp *groupname* === or => chgrp *GID* ===  
(To change the GID of a file or directory. This must be done by SU or the owner, and the owner must belong to the new group.)  
**::groups** = shows all groups  
groups *username* = shows all groups this user belongs to.
- a16.12 **::who** = displays username, login device, login time, remote system name (for remote logins)  
console = screen device used to display system boot and error messages  
pts = pseudo device that represents a login or windows session  
term = serial input/output device
- a16.13 **::last**
- a16.14 **[/etc/passwd]** - maintaining this is an integral part of security. Without an entry here, users cannot login.  
It is divided into the following seven fields:  
=> loginid: x: UID: GID: comment: home directory: login shell  
loginid = user's login name (8 characters max.)  
x = placeholder for password  
UID = a number usually 100-60000. 0-99 for system; 60001 for nobody; 60002 for noaccess; duplicate UIDs are legal, but avoid them.  
{stty -lcase}  
GID = a number usually 100-60000  
comment = First and last name, any additional comment. Also known as the GCOS field for historical reasons.  
home directory = duh  
login shell = /bin/sh; /bin/csh; or /bin/ksh  
[change password requirements]  
> vi /etc/default/login  
passreq=yes (or no)
- a16.17 **[:/etc/shadow]** - contains password information referred to in /etc/passwd. Has the following 8 fields.  
=> loginID:password:lastchg:min:max:warn:inactive:expire  
loginID = username  
password = This field may contain the following entries: a 13-character encrypted user password, the string \*LK\*, which indicates an inaccessible account or the string NP, which indicates no password.  
lastchg = The number of days between Jan. 1, 1970, and the last password modification date.  
min = The minimum number of days required between password changes.  
max. = The maximum numbers of days the password is valid.  
warn = The number of days a user is warned before a password expires.  
inactive = The number of inactive days allowed for that user before the user's account is locked.  
expire = The date when the user account expires. Once exceeded, the user can no longer log in.
- a16.18 **[:/etc/group]** - Only checked at login. A user's primary group is specified by the GID stored in the passwd database. The /etc/group database defines all system groups and specifies any additional groups to which a user belongs. (Users can belong to a primary, and up to 15 secondary groups).  
In addition, while each passwd record specifies both a UID number and user name, it only specifies a user's GID, not the name of the group. The correspondence between a GID and a group name is established by the /etc/group database.  
=> *groupname:password:GID:userlist*  
groupname = Group name, up to 8 characters in length.  
password = This unused field is for a group password.  
GID = Group Identification number.  
Userlist = A comma-separated list of users who belong to this group in addition to their primary group. This is referred to as secondary group membership.
- a16.20 **[:/etc/default]** - Several ASCII files containing variables that specify system defaults are located in this directory.

password file - controls system-wide password aging.  
login file - among other things, helps restrict Superuser access.  
su file - defines the logging of all su attempts by users.

a16.21 **[/etc/default/passwd]** - starting point for password setup.

MAXWEEKS *variable* = Specifies the maximum number of weeks a password is valid before it must be changed for all normal users. If defined as null (the default), only users who have a value for MAX specified in the /etc/shadow file must change their passwords at the specified times.

MINWEEKS *variable* = Specifies the minimum number of weeks between password changes for all normal users. If defined as null (the default), only users who have a value for min specified in the /etc/shadow file are limited as to when they may change their passwords. (The /etc/shadow file takes precedence.)

PASLENGTH *variable* = Specifies a minimum password length for all normal users. Currently, changing this has no effect.

a16.22 **[/etc/default/login]** - Controls su access, among other things.

# CONSOLE=/dev/console = login as anyone, = null, login as root only.

PASSREQ *variable* = If set to YES, (which is the default), users will be prompted. Otherwise, a null password would be valid.

(CONSOLE *variable* = Can be used to specify three conditions for Superuser logins.)

> CONSOLE=/dev/console - login as the Superuser only permitted from the console.

> no entry - logging in as the Superuser is permitted from anywhere.

> CONSOLE= - Logging in as Superuser is not permitted. In this case, the only way to gain Superuser privileges is to log in as a normal user and become Superuser with the su command.

a16.25 **[/etc/default/su]** - Logs all attempts to become su.

SULOG *variable* = Specifies the name of the file where all su attempts to switch to another user are logged. If undefined, su logging is turned off.

CONSOLE *variable* = By default, this variable is ignored because of the comment (#) symbol preceding the entry. However, all su attempts are logged to the console regardless of success or failure. If the value of the CONSOLE variable is defined as /dev/console (by "uncommenting" the entry), all successful attempts to su to root are logged on the console with additional output in the format of the sulog. Unsuccessful attempts are the same as before.

a16.26 **[monitor the sulog]** > more /var/adm/sulog

SU = su command

10/20 = date, time

+ or - = indicates if the command succeeded or not

console = lists the device type

root-sys = name of the user and the switched identity

(This file will keep appending until the system is full, so do the following: > head -100 file1 > file1)

a17.16 [Admintool - password aging]

a18.3 **[Open Boot PROM]** - The open boot programmable read-only memory (OBP PROM) is used on all Sun Sbus type systems.

The OBP PROM monitor is a firmware utility for initializing the system prior to boot. It enables greater functionality than previous type Proms, including the ability to specify booting paths to many more device addresses.

::**banner** = Use the banner command at the ok prompt to identify your system's PROM version number.

ok > banner >> ROM Rev. 2.15 (on line two)

The PROM monitor enables the user or system administrator to customize the environment, and provides an interface to the physical devices attached to the system through buses.

Sparc Bus (Sbus) = the bus on the motherboard.

SCSI Bus = the peripheral bus.

a18.4 ::**probe-scsi** or ::**probe-scsi-all** = [display devices connected to the SCSI bus]

a18.5 **[The SCSI Bus]** - General Information

art/sa135\_18-05.gif

internal disk - Target 3

[a0]

[a1]

a2

internal disk - Target 1

[a0]

a1

a2

•Targeting associates a device driver with a peripheral device. This enables the kernel to identify the device.

•Solaris 2.x provides dynamic allocation of controllers. This enables the system administrator to set unique target numbers, install the peripheral, and the kernel will configure the correct device driver.

•The host adapter numbers are automatically assigned by the operating system based on their on-board or slot location.

### **Default Conventions**

3 - primary internal disk

- 1 - secondary internal disk
- 2 - primary external disk
- 0 - secondary external disk
- 4 - tape drives
- 5 - tape drives
- 6 - CD-Rom

a18.6 **[OBP Revision 1.x Device Names]** - Device names at the PROM level are specified with the boot command using the following format on older machines:

=> *device (ctlr#,target#,file#) path options*

*device* = one of le, sd, or st.

*ctlr#* = the number of the device bus. It is always 0 on 1.x PROMs.

*target#* = the address of the device controller. It is usually 0. For the Ethernet interface, it indicates the host number of the remote host.

*file#* = the partition number of a disk, the file number on a tape, or the file system number over the network.

*path* = the path name of the program you want to execute.

*options* = the arguments to be passed to the program you want to execute.

[How to boot OBP 1.x platform] > b sd (0,3,0) ;; >> ok ;; boot sd (0,3,0)

(n = new mode ("ok" response))

a18.7 **[OBP Revision 2.x Device Names]** - To specify a boot device on a SPARCstation 2 system and above without using the physical device name, use the devalias command to identify possible boot devices.

::**devalias**

>> ok

> devalias

>> screen /iommu@f,e0000000/sbus2f,?..

>> ttyb /obio/zs@o,100000:b

(etc.)

>> ?

>> disk /iommu/sbus.../sd@3,0

>> cdrom /iommu/sbus?/sd@6,0:d

The device alias name is listed on the left side of the output. The disk device alias usually identifies the system's default boot device.

The devalias name for the disk at target address 3 is disk. Since this is the only disk on this system, the default boot device should be set to "disk"

::**boot disk** = how to boot OBP 2.x platform

>> ok ;; > boot disk

a18.8 **[Setting the boot device - PROM Variable Settings]**

::**printenv** => *printenv command* (use to display the default boot device.)

>> boot-device disk

a18.9 ::**setenv** => *setenv command*

Use the *setenv* command to reset PROM settings, *printenv* to verify the new setting, and *reset* to make the change permanent and reboot. In this example, the default boot device is changed from *disk2* to *disk*.

**[change the boot device]**

ok > setenv boot-device net

>> boot-device = disk

ok > printenv boot-device

>> boot-device net disk

ok > **reset**

a18.10 **[boot devices]** - Determine a system's PROM revision level by typing the banner command at the ok prompt.

> **halt**

ok > banner

[how to boot Sun Platforms]

ok > boot cdrom (to boot from a cdrom)

ok > boot (to boot from a harddrive)

a19.3 **[Disk Architecture]**

art/sa135\_19-03.gif

- Disks are composed of several *platters*.

- The platters rotate around a *spindle*.

- The *read/write heads* are moved as a unit by the head actuator arm.

#### A19.4 **[Physical Geometry of the Disk]**

art/sa135\_19-04.gif

a19.5 The Solaris 2.x file system takes advantage of the low-level formatting parameters to improve disk I/O performance.

- A disk drive is comprised of a series of disk platters.
- The smallest units on the platter are ::**sectors** of 512 bytes each.
- Sectors are sections of a ::**track**. The sectors making up a track can be read or written by a given head in one position during a single disk revolution.
- The surfaces are formatted as concentric data tracks, which are completed in one full disk rotation for a given head position. The sum of tracks provided by all the heads at a given position is known as a ::**cylinder**. Because a disk is constantly spinning and because read/write heads move as a unit, the most efficient seeking occurs when the blocks to be read or written are located in a single cylinder.

Data stored on each platter is read from and written to by a series of heads over the surfaces.

#### **[Calculate the size of a disk]**

(512 bytes) x # of sectors per track x # of tracks per cylinder x # of cylinders.

512b x s x t x c = size of disk

#### a19.7 **[Disk ::Partitions ::Slices]**

On Sun systems, disk storage devices are divided into sections called slices (partitions). Slices are groupings of cylinders that enable the system administrator to organize data functionally.

- A disk drive provided by Sun can contain up to eight slices. Each partition is treated by the operating system as a logical disk (a completely separate disk device). They are labeled 0-7 in Solaris 2.x; and are labeled A-H in Solaris 1.x.
- Slice 0 and 1 by default contain root files and swap respectively.
- By definition, slice 2 represents the entire disk. (Don't alter).
- System administrators assign additional slices such as slice 4 for user data files.

#### **[::Swap space]**

Swap space is required for a virtual memory operating system like Solaris 2.x to run.

Installtool will use slice 1 as the default slice allocated to swap.

Sizing of swap space depends on applications. (swap = RAMx2).

#### **[::File Structure]**

The Solaris 2.x computing environment stores data in a logical file hierarchy, which is composed of a number of file systems. The entire file hierarchy is sometimes referred to as the SunOS 5.x or Solaris 2.x file system or file structure.

"file system" = describes a hierarchy of files and directories in a slice, or can describe an entire file tree.

"file structure" = the entire file tree.

#### a19.9 **Slices and File Systems**

art/sa135\_19-09.gif

a19.11 The file structure tree consists of a root (/) file system and a collection of mountable file systems. The mountable file systems are attached at directories to the file system tree with the mount command.

#### The / (root) file system

At the top of the file hierarchy is the root file system, which contains system operating files and directories.

#### The /usr file system

Contains executable commands, system administration utilities, and library routines.

#### The /home or /export/home file system

Contains the users' home directories. This is also referred to as the home file system.

#### The /opt file system

Contains optional, unbundled, and third-party software applications.

#### **Logical Device Name Layout**

art/sa135\_19-12.gif

a19.13 **[File Systems Logical or Device Name]** - Sun systems use the following naming convention to describe the logical device name for a disk.

>> /dev/[r]dsk/c##t##s##

c = controller number

t = target number

d = disk or LUN number

s = slice/partition number

#### **[Controller Number]**

Controller (or interface) numbers such as c0, c1, c2, and so on, are automatically assigned in sequential order to each bus interface.

If your system has a built-in SCSI interface, the OS automatically assigns c0 to it. Controller number c1 would

correspond to a second SCSI controller.

### **[Target Address]**

An external device usually has an address switch located on the rear panel. By default, desktop systems have the first internal disk drive set to target 3. The second internal disk drive is set to target 1.

### **[Disk Number]**

Also known as the logical unit number (LUN), and is always set to d0 for any embedded SCSI device.

### **[Slice (Partition) Number]**

Slice numbers range from 0-7. Disk devices are accessed by their logical device names, and this name must include the slice number. Disks cannot be accessed by just their controller/target/disk designation.

dsk = block special

fsck = raw

### **[Physical Names]**

Logical device names are found in the /dev directory and are symbolically linked to their corresponding physical device names in the /devices directory.

**[/dev/dsk]** - contains the logical names of the disk drives.

a19.15 **::mount** - a local file system is attached to the root file structure with mount.

The directory on which a file system is mounted to the root file hierarchy is called a mount point. The mount point must exist before the mount command is issued. Only root can mount a file system, and can make it read only.

=> mount *file\_system mount\_point*

> mount /dev/dsk/c0t3d0s4 /export/home (this is how you use the mount command)

> mount /dev/dsk/c0t6d0s2 /cdrom

a19.17 > mount = displays currently mounted file systems

### **[format]**

*file\_system (mount\_point) on logical\_device\_name* = identifies the file system mounted on the disk partition.

*permissions* = identifies the general file system permissions.

*on date and time* = identifies the date and time the file system was mounted.

a19.18 **[mounting file systems]**

Files for mounting - Mounting file systems from the command line creates an "attachment" known as a file handle.

The file handle is kept in memory and is lost when the system is rebooted.

[/etc/vfstab] - If this doesn't exist, you must manually remount it at boot time.

Maintains all information necessary to mount file systems at boot time.

[/etc/mnttab] - current time

Contains a record of all mounted files systems. The mount command and other system routines read this file to identify system resources.

a19.20 **::df** = "disk free" (disk space free)

=> df [-k] *directory*

-k = Displays usage in Kbytes and subtracts the space reserved by the operating system from the amount of available space. (the head room).

About 10% of disk capacity is reserved for file system proficiency. This is not reflected in the df -k output.

a19.22 **::du** = The du command is used to display the number of disk blocks (512 bytes) used by directories and files.

=> du [-a] [-s] [-k] *directory*

-a = display the number of blocks used by all files and directories within the specified directory hierarchy.

-k = in Kbytes

-s = display on a summary

a19.24 **::quot** = displays how much disk space in Kbytes is used by users.

=> quot [-af] [*filesystem . . .*]

-a = report on all mounted file systems.

-f = include number of files.

a20.6 **[Solaris installation & software groupings]**

Installation software is composed of three parts:

- packages

- clusters

- configurations

art/sa135\_20-06.gif

a20.7 **[Packages]**

A software package is a group of files and directories. For example, all files and directories related to the on-line manual (man) pages are in a package called SUNWdoc.

Creating a software package is the standard way to deliver bundles and unbundled software. Packages are administered using the package administration commands and are generally identified by a SUNWxxx naming convention.

For example, the OpenWindows software package names used with the package administration commands are: SUNWxwacx, SUNWoldcv, SUNWoladd, SUNWolimt, etc.

### **[Software Clusters]**

During installation, the software packages are grouped into software clusters, which are logical collections of software packages. For example, the System and Network Administration cluster includes the following packages:

- System and Network Administration Applications
- System and Network Administration Framework
- System and Network Administration Root

Some clusters include only one package. For example, the On-line Manual Pages cluster contains one package—the on-line Manual Pages.

### a20.11 **[Hardware Requirements]**

To use the Installtool utility to install and run the Solaris 2.x software locally, the system must meet the following requirements:

- It must be based on a SPARC or an Intel system.
- It must have at least 300 megabytes of disk space.
- It must have a minimum of 16 meg of memory. 32 megs for OpenWindows.
- It must include a CD-Rom.

[sysidtool - System Identification Information]

During the first part of the installation process, you are prompted to supply the system name and basic network information. Fill out the installation worksheet summarizing the system identification.

a20.12 **::halt** - Shuts down the system to an idle state. Syncs the file system.

[stop-a] - halts processes without a sync to the disk.

### a20.13 **[How to Install Solaris 2.x]**

Determine the system's PROM revision level. >>ok > banner

>boot cdrom

[host name] - 64 character max. (Do not start with #), no spaces, lower case

- located in:

>> /etc/inet/hosts

>> /etc/nodename

>> /etc/hostname/le0 or /etc/hostname/ie0

>> /etc/net/tic\*/hosts (3 files)

a20.34 **[Preserve Data]** - Preserving an 'overlap' slice will not preserve any data within it. The mount point name must be explicitly set.

a20.37 **[Auto-layout]** - choose auto-layout. You can change later. Don't change swap and overlap.

a20.38 **[Remote Mounts]** - Test manually before you put in "vfstab". Put it at the end.

a20.42 **[/var/sadm/install\_data/install\_log]** - This file shows the results of an installation.

a21.8 **[::Networking Files]** - This section describes the files that are used to define a system's characteristics in terms of network configuration. These files are created automatically during the system configuration phase of installing a system.

**[::/etc/inet/hosts]** - each Internet address has a corresponding host name. The /etc/inet/hosts file associates IP addresses with host names. This means users and system administrators can refer to systems by their host names rather than their IP addresses when using networking software. The /etc/hosts file is a symbolic link to this file.

The following hosts file defines the IP address and host name for the local system (venus) and two systems on the network.

> Internet host table

127.0.0.1 localhost (loop back)

129.150.212.16 *venus* loghost(this is an alias - you can have more than one name)

129.150.212.11 *mars*

129.150.212.12 *jupiter*

Network 127 is reserved for the local host network number. The local host address is available so that the local system can run network software without a network, if necessary.

The host name can be followed by one or more aliases. Aliases permit the host to be known on the network by additional names. Sometimes, an alias identifies the host to be the provider of a special network-wide service (for example, the main mail provider is aliased as mailhost).

a21.9 **[::/etc/nodename]** - (cat -n) contains the system's host name.

> cat /etc/nodename

>> *venus*

**[::/etc/hostname.le0]** - identifies the Ethernet interface, such as le0, to be configured at boot up and contains the host name or the host's IP address.

> cat /etc/hostname.le0

>> venus

a21.10 **::rlogin** - enables a remote login session if you have an account set up on the remote system.

=> rlogin *hostname* -l *username*

Ex: > rlogin *fusion* -l *corries*

-l = specify a different user ID

a21.11 **::rsh** - used to execute a program on a remote system.

=> rsh [-l *username*] *machinename* *command*

Ex: > rsh fusion showrev

a21.12 **::rcp** - enables you to copy files or directories to and from another machine.

**[How to copy files across the network]**

=> rcp source-file machinename:destination\_file =>

**[from the local directory to remote host]** > rcp scot.id river2:/tmp

=> rcp machinename:source\_file destination\_file <=

**[from remote host to (local) /tmp]** > rcp river2:/tmp/scot.id /tmp

**[remotely copy directories with the -r option]** > rcp -r ~/perm river2:/tmp

*[Files in /tmp are cleared out each time the machine is rebooted.]*

art/sa135\_21-14.gif

a21.15 **[Security]** - Several files are important to network security when using the rlogin, rsh, and rcp commands.

*[/etc/passwd]* - This is the first file the system looks at, and if the password entry also includes an entry in */etc/shadow*, the user must supply a password upon login.

*[/etc/hosts.equiv]* - can identify remote machines as trusted hosts. The host name must also be in the */etc/inet/hosts* file on the remote machine.

This is useful for sites where it is common for users to have accounts on different machines, and it eliminates the security risk of sending ASCII passwords across the network. If the remote command is trying to execute as the Superuser, the */etc/hosts.equiv* test is skipped.

This file must be created. It does not exist by default.

a21.17 **[:/etc/hosts.equiv and .rhosts]**

While the */etc/hosts.equiv* and *.rhosts* files have the same format, the same entries in each file have different effects. (Neither file exists by default; they must be created.)

•Both files are formatted as a list of one-line entries

*hostname*

*hostname username*

•If only the hostname is used, then users from the named host are trusted. That is, every user from the named host is a trusted user, provided they are known to the remote system.

•If both hostname and username are used, then only the named user from the named host can access the system.

"+" can be used in place of either hostname or username to match any known host or user. For example, the + entry by itself enables any user from any known host to access the remote system as a user with the same username, without needing a password.

•The host names in the */etc/hosts.equiv* and *.rhosts* files must be the official name of the host, not one of its nicknames.

a21.18 **[Remote Access Authentication - using the /etc/hosts.equiv file]**

Typically, the */etc/hosts.equiv* file contains a list of host names (one per line) or a single plus sign (+), indicating that all hosts are trusted hosts on your system.

Examples:

> + (all hosts are trusted)

> fusion

> speedy (two hosts are trusted)

> fusion corries (specific user from a host)

This specifies that the user corries can access the remote host from the host pluto as any local user. Under most circumstances, *this is unsecure*.

The above entry would permit corries to use any of the following commands to access river1 without requiring a password.

> rlogin river1

> rlogin river1 -l lister

> rlogin river1 -l kryten

a21.19 **[\$HOME/.rhosts {personal} file]**

If the */etc/hosts.equiv* test fails, the next step is to check for a *.rhosts* file in the user's home directory on the remote host.

The default is no.rhosts file

The most conventional usage is to explicitly name a host. This enables users from the remote machine speedy to access this system as the user whose home directory contains the .rhosts file. It assumes that the login name is the same on both hosts.

> speedy

The following is an example of a user's .rhosts file, which allows users to rlogin from any known system on the network, without supplying a password.

> +

As with the /etc/hosts.equiv file, any host name in a user's .rhosts file can be followed by a login name.

> fusion carries

If this .rhosts file is in the user's *doej* home directory on a system, the user *carries* can access this machine from the host fusion, as the user *doej*.

a21.22 **::rusers** - used to identify who is logged in on the network. Press C]-c to get the prompt back.

=> rusers [-options] *hostname*

-a = gives a report for all systems, even if no users are logged in.

-l = Displays the long listing (such as the who command)

Ex: > rusers -la or > rusers speedy

a21.24 **::spray** - Because the ping command uses a low-level protocol, it is still possible for a host to respond to a request even though it is not up and running. Unlike the ping command, /usr/sbin/spray uses the higher-level protocols. The spray command is typically used to test the response of a machine on the network over a period of time.

Ex: > spray river1

a21.25 **::netstat** - displays the status of various network-related data structures. The form shown in the example (using the -i option) shows the usage of the network interfaces on your machine.

> netstat -i

```
>> Name      Mtu   Net/Dest      Address      Ipkts   Ierrs   Opkts   Oerrs   Collis   Queue
      lo0      8232  loopback      localhost     769     0       769     0       0        0
      le0      1500  neptune       venus        211053  0       18405   0       676      0
```

Name = The network interface. lo0 is the loopback interface, used to test the network protocols in software.

MTU = The maximum transmission unit (MTU). The size, in bytes, of the data of the largest packet this interface supports. Ethernet has an MTU of 1,500 bytes, and fiber optic data distribution interface (FDDI) is 4,428 bytes.

The loopback network has a MTU of 8,232.

Net/Dest = The name of the network destination.

Address = The host name. If you use the -n option, you also see the Internet address.

Ipkt/Ierrs = Shows the number of input packets and errors since the interface was configured.

Opkts/Oerrs = Shows the number of output packets and errors since the interface was configured.

Collis = The number of collisions on this interface. To calculate a % rate, use (Collis/Opkts)\*100=collision rate. An excellent collision rate is 0-2%, fair is 3-5%, and poor is over 5%.

Queue = The number of packets awaiting transmission at the interface. This is almost always zero.

a22.3 **[::Print Service Features]**

Printing Services

- Daemons that monitor print requests.
- A hierarchy of configuration files in the /etc/lp directory.
- A set of lp administration and print commands.
- A GUI interface (Admintool) for printers.

Features

- Provides a variety of printer service functions.
- Supports printing forms, print wheels, character sets.
- Includes PostScript filters.
- Supports a wide range of printers.
- Interoperates with SunOS 4.1.x (BSD-based) print service.
- Provides flexible printer management, including: print jobs, printer classes, selected printer access.

a22.4 **[Print Functions]**

Queuing

Tracking

Fault Notification

Initialization

Filtering

a22.5 **[Forms and character support]**

Print forms = forms with text and graphics--such as an invoice, blank check, or letterhead---and are used instead

of blank paper.

Software character sets that come preprogrammed with the printer, such as non-Postscript printer, must be predefined by the system administrator as a specific font style. Then the user prints a file with the font style as an option.

Selected character sets may be available with a supported printer type, and are listed in the terminfo database. [terminfo] - database consists of a series of files that describe control sequences for initializing printers and terminals.

The LP service will enable you to identify printer types, file content types, filters, and interface programs that are needed to set up a printing environment.

#### a22.6 **[Content Types]**

Every print request consists of at least one file containing information with a particular format, which is called a content type.

Every printer must be defined with a printer type and at least one content type. The print service uses this configuration information to match a print request to a printer that accepts that type of print request.

[Matching Print Requests to Printers]

If you have a PostScript printer, specify that the content type is PostScript. The only time a user needs to specify a content type when printing a file is if the file needs special filtering so the printer knows what filtering needs to be done.

#### a22.7 **[Print Filters]**

The Solaris 2.x release provides a default set of PostScript filters that are automatically installed when a PostScript printer is configured using the Admintool.

PostScript filter information is stored in several places:

/usr/lib/lp/postscript = default PostScript filters.

/etc/lp/fd = set of print filter descriptor files

/etc/lp/filter.table = a print filter lookup table of these downloaded filters

#### a22.8 **[Printer Types]**

The printer type is used to identify the terminfo database entry containing the control sequences to initialize the printer. This database contains a large number of entries which, in most cases, means there is no need to create additional printer entries.

Supported printer types include PS(PostScript), PSR(PostScript Reverse), and non-PostScript types such as daisy and diablo.

**[Checking for Defined Printer Types]**

To check if a printer entry exists in the terminfo database, you can use the User Accounts, Printers and Mail Administration guide to verify that your printer is listed in the Frequently Used PostScript and Non-PostScript Printers tables.

You can also list the contents of the /usr/share/lib/terminfo subdirectories.

```
> ls /usr/share/lib/terminfo /e
```

The terminfo entry has a directory name with the same initial letter or digit as the abbreviation of the printer.

#### a22.9 **[Interface Programs]**

Interface programs are usually shell scripts used by the print service to set certain default printer settings.

For example, to override the Admintool settings and turn off the banner page option, modify the /etc/lp/interfaces/printer\_name script on the print server by changing the nobanner line from:

```
>> nobanner="no"
```

```
> nobanner="yes"
```

This script is a copy of the standard initialization script, /usr/lib/lp/model/standard, which takes the initialization information from the printer type entry in the terminfo database.

The default printer settings are described in the User Accounts, Printers and Mail Administration guide.

#### a22.10 **[Local and Remote Printers]** - a local printer is a printer that is attached directly to the system. A remote printer is a printer that a user accesses over the network.

The Solaris 2.x release provides a heterogeneous printing environment.

#### a22.12 **[Local Printing Model]** - Overview of the lpsched daemon

lpsched = daemon, which is the print service scheduler, is responsible for tracking all local print requests and updating the lp database with printer information.

When a print job is submitted with the lp command, the scheduler places a copy of the request in the /var/spool/lp/requests directory.

Next, the lpsched daemon matches the request contents to the printer contents and identifies a filter, if a conversion is necessary. When filtering is complete, the scheduler waits for the printer to become available; then it runs the interface program to initialize the printer and downloads the request to the port to which the printer is attached.

art/sa135\_22-12.gif

a22.13 **[Remote Printing Model]**

Almost identical to local printing. The main difference is that the lpsched daemon hands the print request to the lpNet daemon rather than downloading it to a local printer. The lpNet daemon is capable of establishing a connection with a listen or lpd process running on the print server.

art/sa135\_22-13.gif

a22.14 **[Print Servers and the lpNet daemon]**

Each print client and server has at least one lpNet daemon that schedules network print requests.

In the case of a Solaris 2.x print client, the lpNet process connects to the server's listen port monitor, which connects the incoming lpNet request to a local lpNet process. Once the request has been queued by the local lpNet process, it is sent to the printer by the lpsched daemon.

art/sa135\_22-14.gif

a22.15 **[Print Services]**

Configuring printer services involves the following three things:

Setting up the printer

1. physically setting it up, and adjusting switches.

Setting up the Print Server

A print server is a system with a printer connected to it and is configured to provide access to the local printer using Admintool.

Setting up the Print Client

A print client is a system that uses a print server for printing and is configured to provide access to the remote printer using Admintool.

a22.16 **[Print Server Requirements]**

The system to which the printer is attached is the print server for that printer. Any networked system that meets the requirements below can be a print server.

Print server minimum requirements:

- Spooling directory space of 8 megabytes (or more).  
Spooling space (/var/spool/lp) is the most important factor. For an active print server, 60-100 megs is common. Files are also generated in the /etc/lp directory and so available room in the root directory (if /var is separate) should be taken into account.
  - A hard disk.
  - Greater than 16 megs of ram is recommended.
  - Swap space of 20 megs or more.

a22.18 **[Network Access]**

If your network is not running a name service, each print client's system name and Internet address must be in the print server's /etc/inet/hosts file before setting up the print servers and print clients. Verify.

Rome 155.60.20.25 (name=lp-wyse)

**Basic LP commands**

BSD    SVR4

lpr      lp

lpq      lpstat

lprm     cancel

> banner This is a test | lp

> lpstat -T

a22.18 **[How to add a local printer]** - Use Admintool; Browse Printers; Add Local Printer or Access.

a23.3 **[Basic lp Commands]**

Command    Description

lp            Sends file to a printer

lpstat       Displays print service status

cancel       Cancels print requests

accept       Enables queuing of print requests

reject       Prevents queuing of further print requests

enable       Enables printer to print requests

disable      Disables printer from printing requests

lpmove       Moves print requests

lpadmin      Performs various administrator tasks

a23.4 **::lp** - Use to print ASCII or text files, do not use to print files created in applications like Framemaker. The function of the lp command is to queue data for printing.

=> lp -options filename/s

-d printername = specifies a specific printer to print your file

-n num. = specifies to print num. of copies of the file.

[How to print a file]

```
> lp ===  
> lp -n 2 ===  
> lp -d riv106 ===
```

a23.5 **::lpstat** - use the lpstat to display the user's print queue

=> lpstat -options

-a = reports whether print destinations are accepting requests

-d = displays the name of the default printer

-o = displays the status of all output requests on printers

-p = displays the idle or busy status and availability of all printers

-s = determines what printers are configured for the system on which you are working

-t = displays all status information output by -s, plus the acceptance and idle or busy status of all printers

[Ex: display the default printer]

```
> lpstat -d
```

a23.7 **::cancel** - use to cancel a specific printer request waiting in the queue or to cancel the print request currently printing.

=> cancel request-ID printer

=> cancel -u user printer

[How to cancel print jobs]

```
> lpstat -o
```

```
>> post-32 clarkc ?.
```

```
> cancel post-32
```

```
> cancel -u student1 staffp
```

[How to cancel a print job that is currently printing]

```
> cancel post
```

```
> LPRM - job# it
```

a23.8 **[Designating a default destination]**

A system administrator can designate a printer or a class of printers as the system-wide default destination for print requests using the lpadmin command.

Ex: > lpadmin -d it

```
> lpstat -d
```

```
>> system default destination: it
```

(Can also be done using Admintool)

[How to set a user default printer variable] - Individual users can set their own default printer by setting the LPDEST environment variable with the name of that printer or class.

(in BO or KO shell) > LPDEST=it;export LPDEST

(in C shell) > setenv LPDEST it

a23.9 **[Using Printer Classes]**

Class = a named group of printers created with the lpadmin command. Once created, a class can be used as the destination of print requests enabling the print service to automatically select an available printer within the class that has a matching content type.

Class Criteria

Printer classes can be defined using different criteria:

- Based on printer type (all PostScript printers)
- Based on location (first floor, tenth floor, etc.)
- Based on workgroup or department (Accounting or Engineering)

Printer Priority within a Class

You can create a class of printers to ensure the printers are accessed in a particular order, because the print service always checks for printer availability, using the order in which the printers were added to the class. For example, if you have a high-speed printer, add it to the class first so it can handle as many print requests as possible. These rules also apply to printer usage order.

a23.10 **[Create a class of printers]**

```
> lpadmin -p riv107 -c f1
```

```
> lpadmin -p riv108 -c f1
```

```
> accept f1
```

[Check the status of a printer class]

```
> lpstat -t
```

```
? >> f1 accepting requests since Tue Jun 8 11:26 ?
```

[Print to a class of printers]

```
> lp -d f1 worldmap.ps
```

- >> request id is f1-30 (1 file (s) )
- a23.11 **[Managing the Queue]**  
 > lpc status  
Putting jobs on hold  
 lp -l print-request -H keyword  
 hold = hold the specified job  
 resume = resume a job previously held  
 immediate = move the specified job to the top of the queue (Only root can do)  
Examples:  
*Put on hold*  
 > lp -i riv107 -H hold  
 > lpstat -o spock  
 >> ? being held  
 Resume a job  
 > lp -i riv107 -H resume  
 Place a job at the top of the queue  
 > lp -i riv096 -H immediate  
 Remove all jobs to a specific printer  
 > lpc clean riv105  
 Print a test to a printer  
 > lp -d it
- a23.12 Solaris 2.x enables users to submit print requests at various priorities. Priorities range from 0 (highest) to 39 (lowest). The default priority for all users is 20.  
 [Place a job at a high priority]  
 > lp -d riv110 -q 0 fastfile  
 [Place a job at a low priority]  
 > lp -d riv110 -q 30 bigfile
- a23.14 **[To take a printer out of service]**  
 •Reject additional print requests on the printer to be made unavailable  
 •Move or cancel any requests that are currently queued to the printer.
- a23.15 **[Move print jobs]**  
 Use the lpmove command to move selected print requests, from one printer or printer class to another.  
 > reject -r "riv095 is down for repairs" riv095  
 (This step notifies users why the printer is not accepting requests)  
 [checking to see what needs to be moved] > lpstat -o  
 [verify another printer is accepting requests] > lpstat -a riv095
- a23.16 **[Move specific or all print requests]**  
 > lpmove riv095 riv096 or > lpmove riv095-11 riv095-12 riv096  
 (Use accept, once the unavailable printer is available again) > accept riv096
- a23.17 **[Troubleshoot a printer]**  
**[Check the status of the queues]**  
 > lpstat -o  
**[Stop and restart daemons]**  
 > /etc/init.d/lp stop  
 > /etc/init.d/lp start  
 The print service scheduler is automatically started by the /etc/init.d/lp script when the system comes up.  
 [Remove a printer configuration manually]  
Remove the print queue  
 > rm -r /var/spool/lp/requests/hostname/\*  
 (All files in the queue are lost)  
Remove the printer configuration  
 > lpadmin -x printname  
 Stop and restart daemons  
 Setup the printer through Admintool or command line
- aB.2 **[Configuring Printers]**  
**[How to configure a local printer]**  
 1.Become root, and ensure the /usr/lib directory is in your command search path.  
 2.Change ownership and set permissions on the printer's serial port.  
 > chown lp /dev/term/a  
 > chown 600 /dev/term/a

3. Use `lpadmin` to add the printer and associate it with a printer port.
  - > `lpadmin -p riv202 -v /dev/term/a`
  - p = indicates the printer name
  - v = indicates the device used by the printer
 This command also registers the printer name with the print service. From now on, use this name to identify this printer.
4. Associate the printer with a content type.
  - > `lpadmin -p riv202 -I simple`
  - I = indicates the content type
 If the content type is not specified, the system uses the simple type, which means the printer can only handle ASCII contents.
5. Set the line parameters appropriate for the printer.
  - > `lpadmin -p riv202 -o "stty=1200 evenp"`
6. Associate the printer with a printer type, if necessary. This enables the interface program to perform a better initialization of the printer before downloading every request.
  - > `lpadmin -p riv202 -T proprinter`
7. Enable the printer to accept requests and enable the printer.
  - > `accept riv202`
  - > `enable riv202`

aB.4 **[::Configure a local postscript printer]**

The `lpfilters` installs filters on the system by updating the filter table with the filter name and description file.

1. Become root, and ensure the `/usr/lib` directory is in your command search path.
2. Change ownership and set permissions on the printer's serial port.
  - > `chown lp /dev/term/a`
  - > `chown 600 /dev/term/a`
3. Use `lpadmin` to add the printer and associate it with a printer port.
  - > `lpadmin -p riv210 -v /dev/term/a`
  - p = indicates the printer name
  - v = indicates the device used by the printer
 This command also registers the printer name with the print service. From now on, use this name to identify this printer.
4. Associate the printer with a content and printer type.
  - > `lpadmin -p riv210 -I PS -T PS`
5. Use the `lpfilter` command to register the PostScript filters.
  - > `cd /etc/lp/fd`
  - > `lpfilter -f download -F download.fd`
  - > `lpfilter -f dpost -F dpost.fd`
  - > `lpfilter -f postdaisy -F postdaisy.fd`
  - > `lpfilter -f postdmd -F postdmd.fd`
  - > `lpfilter -f postio -F postio.fd`
  - > `lpfilter -f postior -F postior.fd`
  - > `lpfilter -f postmd -F postmd.fd`
  - > `lpfilter -f postlplot -F postlplot.fd`
  - > `lpfilter -f postprint -F postprint.fd`
  - > `lpfilter -f postreverse -F postreverse.fd`
  - > `lpfilter -f posttek -F posttek.fd`
  - > `lpfilter -f pr -F pr.fd`
6. Enable the printer to accept requests and enable the printer.
  - > `accept riv210`
  - > `enable riv210`

aB.6 **[Remove a local printer]**

1. Suspend the queuing of further requests with the `reject` command.
  - > `reject -r "printer riv2000 is down" riv2000`
  - r = indicates a reason message. This message is displayed when users issue the `lpstat` command.
2. Use the `disable` command to disable the printer or to stop printing.
  - > `disable -W -r "printer dotted is down" riv2000`
  - W = wait until the request currently printing is finished before disabling the printer. This option is ignored for remote printers.
  - c = Cancel the currently printing request (not used in the example). This request is restarted when the printer is enabled again. The request is ignored when applied to remote printers.



[User commands] - in /usr/bin

[Administration commands] - in /usr/lib

Command      Description

lpssystem(1M) registers remote systems with print service

lpmove(1M) moves requests between destinations

lpusers(1M) changes user print priority settings

lpfilter(1M) registers filter definitions with print service

aB.14 **[Configuration Files]**

<u>File Name</u>	<u>Type</u>	<u>Description</u>
/etc/lp/Systems	file	list of remote hosts registered with print service
/etc/lp/default	file	contains name of system-wide default destination
/usr/lib/lp/postscript	directory	contains filter description files
/usr/lp/filter.table	file	printer filter lookup table
/etc/lp/logs	symlink	symbolic link to /var/lp/logs
/etc/lp/printers	directory	contains one subdirectory for each configured printer
/etc/lp/printers/riv2000	file	configuration file for printer pname
/var/lp/logs	directory	print service log files
/var/spool/lp/SCHEDLOCK	file	lpsched lock file prevents more than one instance of lpsched process
/var/spool/lp/system/pstatus	file	contains the current status of print system
/var/spool/lp/tmp	directory	the spooling directory

a24.3 **::ps** - processes currently running

=> ps *-options*

-e = print information about every process on the system

-f = generate a full listing

>>> ... /etc/init = main process

>>> ... fsflush = virtual memory management

[zombies] > ps -es (see defunct)

> wait zombies p.i.d. (at least 5 times)

a24.4 **::kill** - usually used to terminate a process on the system.

=> kill *-signal PID(s)*

[kill signals] - There are currently 44 signals defined in Solaris. Each signal is associated with a number and a name. Kill, without a signal specified will default to signal 15(SIGTERM). This usually causes the process to terminate.

a24.5 **::at** - executes a command or script at a specified time. The command only executes once, unlike entries in a crontab file, which are executed over and over again at the specified day and time.

=> at [-m] [-r *job*] time [date]

-m = sends mail to the user after the job is finished

-r = removes specified job previously scheduled

time = time may be specified in a variety of ways, including h, hh, and hh:mm. A 24-hour clock is assumed unless am or pm (case insensitive) is used as a suffix to time.

date = (optional) can be specified as either a month followed by a day or the day of the week.

The /etc/cron.d/at.deny file identifies users who cannot use the at command.

(at, allow file - if any non root user is in it, root must be in the file)

run a shell script > at -f /absolute path to script

a24.6 **[specify time and command to index the man pages]**

> at 20:45

> catman -w

>C] -d

**[look at the queue]** > atq

**[remove a job from the queue]**

> at -r 730001010.a

**[file system state in superblock is wrong]**

clean byte

fs -properly mounted

cannot read block (Kirk McKusick)

a24.7 **::crontab** - use > crontab -l to view the contents of your crontab file. (must be root)

•The cron daemon is started when the system boots and runs continuously.

•The cron daemon reads the crontab files in the /var/spool/cron/crontabs directory.

•The commands are executed at regularly specified times. The time is specified in military time.

a24.12 **[Six fields of cron]**

<i>min after the hour</i>	<i>hour</i>	<i>day of the month</i>	<i>month</i>	<i>day of the week</i>	<i>command</i>
0-61	0-23	1-31	1-12	0-6 (starts Sunday)	/etc/backup/?

{2,4,6,8 ; 10,30,45,55; \* = all values}

The first five fields are integer patterns with the following format:

n = matches if field value is n

n,p,q = matches if field value is n,p, or q

n-p = matches if field has values between n and p inclusive

\* = always matches

Ex: > crontab -l

>> #

>> 0 2 \* \* 0,4 /etc/cron.d/logchecker

>> 1 2 \* \* \* [ -x /usr/sbin/rtc ] && {reverse Boolean logic} /usr/sbin/rtc -c > /dev/null 2>&1

{Standard error should go where standard output goes}

> crontab -l >cronfile (lists current jobs now)

output = 2> /dev/null}

errors = 1 > &2

#### a24.8 [root crontab file]

Two important entries in the root crontab file are:

- /etc/cron.d/logchecker (file) - determines if the cron log file /var/cron/log has exceeded the maximum file size, and if so, copies the contents to the /var/cron/olog file.

- /usr/lib/newslog - cleans out log files generated by the syslog daemon in the /var/adm directory.

(shows ASCII strings in binary)

> file \#12345 ? binary

> strings \#12345 (checks files)

[add this to cron file]

> find / -name core -exec rm {} \; 2 > /dev/null

> vi scot.cronfile (put anywhere) date > /dev/console

> crontab scot.cronfile

#### a24.9 [Controlling crontab access]

The two files that control access to the cron utility are:

- [:/etc/cron.d/cron.allow] - root must be in this

- [:/etc/cron.d/cron.deny] - ignored if allow exists (does not exist by default)

By default, the cron.deny file prohibits crontab use from the following system users:

daemon, bin, smtp, nuucp, listen, nobody, noaccess.

#### a24.10 The crontab command enables you to edit, list, and remove a crontab file.

It stores input in: /var/spool/cron/crontabs/username(s)

Superuser's crontab file is: /var/spool/crontabs/root

> crontab /cronfile

#### a24.12 [edit a user's crontab file, How to]

Tell cron which editor to use.

> EDITOR=vi; export EDITOR

Edit the crontab file to say you are done.

> crontab -e

View the current crontab file

> crontab -l

Caution - If you mistakenly type crontab without any arguments, use the interrupt character C]-c to exit. Do not use C]-d. This will remove all entries in your crontab file.

#### a25.3 [Command-line Mail]

1.Become Superuser

2.Edit the /etc/hosts file to include an entry for each host on the network. (155.60.20.34 corries)

3.Add an entry for the mailhost (155.60.20.4 designate\_host mailhost)

4.This completes a mail client. On the server, copy main.cf to sendmail.cf.

> cd /etc/mail

> cp main.cf sendmail.cf

This completes a mail server for a lan. All of the above must be done before setting up a mail server.

#### a25.4 ::mailx => mailx corries@fusion

Each user has a mailbox file in which to receive mail. Usually in /var/mail/username.

[How to read your mail] > mailx

#### a25.5 [How to send mail]

1.Specify the username@machinename as an argument to the mailx command.

2.Type the subject of your mail and type return.

3.Type the text of your message, and press the . (period) key or C]-d on a line by itself to signal the end of the message.

To cancel while composing it, type C]-c. The message will go to a dead.letter file in the home directory.

a25.6 **[How to read mail]**

> mailx

>> ? hit return, again for next message, etc.

Scrolling:

>> ? z+ = next screen

>> ? z- = previous screen

>> ? h = redisplay the list of mail headers at any time

a25.7 **[The mail header]** - The output from the mailx command is called the mail header, it displays:

- the version of mailx

- the timestamp

- a question mark (?) for help with mail

- the location of the inbound mail box

**[The mail box]**

- the total number of messages in the mail box

- the number of new messages

- the status (new or unread) for each message

- the number of the message in the order received

- the sender (comment field defined in /etc/passwd)

- the time and date the message was sent

- the size of the message (# of lines and # of characters)

- the subject of the message

**[Status]**

N = new message

R = message that has been read

U = letter was new, but was not read before quitting the mailx program.

> = current message

a25.8 **[Deleting messages]**

>> ? d

>> ? d *number*

>> ? d 1 3 5

>> ? d 1-9

**[Undeleting messages]**

>> ? u *number*

**[Reply]**

>> ? r (R to reply to everyone who received the current message)

>> ? R *number*

a25.9 **[Print Command]**

>> ? |*number* |p

a25.10 **[Quit]** >> ? q

**[Access messages stored in your mbox file]**

> mailx -f

a25.11 **[Send a file]**

> mailx jonesk@orion < fusion

> mailx -s "Dante Information" jonesk@orion < fusion

-s = adds a subject line

a25.12 **[Mailx Alias (Distribution list)]**

=> mailx *aliasname*

Add mail aliases to your .mailrc file, which is located in your home (or login) directory. The .mailrc file is used to store private mail aliases and variables relating to the mail program. Or, as root, you can set up public mail aliases in the /etc/aliases file.

You can use any editor to create or add mail aliases to the .mailrc file. Create one mail alias per line, and then save the changes.

Ex: > vi .mailrc

> alias aliasname jonesk@orion jervinc@saturn youtooa@hertzu

a25.13 **[Using Tilde Commands]** - used while writing a letter.

(If you want to use a literal ~ character in a letter, type two tildes in succession, only one will be displayed.)

Command	Function
~!	Escapes to a shell command
~.	Simulates pressing the C]-d to mark EOF
~?	Lists a summary of tilde commands
~b username	adds user name/s to the bcc: list
~c username	adds user name/s to the cc: list
~f number	forwards the specified letter. Valid only when sending a message while reading mail
~h	prompts for header lines: Subject, To, Cc, and Bcc.
~r filename	reads in the text from the specified file
~t name	adds the specified name/s to the To list
~p	prints the message being entered to the screen
~s string	changes the subject line to string
~m number	inserts text from the specified letter into the current letter. Valid only when sending a message while reading mail.
~x	aborts the composition and sending of the email message, totally discards the message.
~v	dive in

#### a25.14 [Transporting Binary Files]

<u>Sender</u>	<u>Results</u>	
1. Compress file: > compress <u>bin.file</u>	<u>bin.file</u> =>	<u>bin.file.Z</u>
2. Encode file and mail: > uuencode bin.file.Z bin.file.Z \   mailx <u>user@host</u>	bin.file.Z	/var/mail/usr
<u>Recipient</u>	<u>Results</u>	
3. Save encoded file from mail box to new file: > mailx ? s bin.file.uu	/var/mail/usr	bin.file.uu
4. Decode encoded file: > uudecode bin.file.uu	bin.file.uu	bin.file.Z
5. Uncompress compressed file: > uncompress bin.file.Z 1.> cp /usr/bin/lis /tmp/tstls 2.> compress -v /tmp/tstls.Z	bin.file.Z	bin.file

a25.15 **::compress** - used to compress files using a special format to reduce the size of the file (by 20%-80%)

Ex: > compress -v bin.file  
> uncompress /tmp/tstls  
.Z = compress, (use this)  
.z = pack (file must be 2100 bytes)

a25.16 **::uuencode** - used to create an ASCII-encoded representation of a binary file. The resulting encoded file is always larger than the original (compressed) file.

The following example encodes the file bin.file.Z, and uses a pipe to mail it to a user.

Ex: > uuencode bin.file.Z bin.file.Z | mailx user@host

The uuencode command requires two arguments-the source file and the file name to be stored in the mail message. The latter is the file name of the file once it is decoded with the uudecode command.

If you want to include a subject line when mailing the binary file, use mailx with the -s option. The example below encodes the file bin.file.Z and uses a pipe to mail it to the user. The -s option enables bin.file.uu to be displayed as the subject line.

Ex: > uuencode bin.file.Z bin.file.Z |mailx -s bin.file.uu user@host

#### a25.17 [Transporting binary files]

Sending binary files through email, there is a high risk of data corruption.

Encoding the binary file ensures that the data is not corrupted during transport. Compressing the file before it is sent via email helps to prevent overflowing the recipient's mailbox.

The procedure for using email to transport binary files is as follows:

- 1.sender compress the binary file
- 2.sender encode the binary file and send it to the recipient by way of email.
- 3.recipient saves the message containing the encoded file.
- 4.recipient decodes the file the message was saved to.
- 5.recipient uncompresses the file

#### a25.18 [Retrieve a compressed and encoded file sent through email]

- 1.Save the mail message with the encoded file to a new file. (The file is called the binary.file.uu in this example.)

2. Decode the file using the uudecode command.

```
> uudecode bin.file.uu
```

After the decoding process is completed, you have two files-bin.file.uu, created in step 1, and the compressed file bin.file.Z, created by uudecode.

The .Z suffix means that this (decoded) file is compressed and must be uncompressed.

3. Uncompress the previously encoded file using the uncompress command.

```
> uncompress bin.file.Z
```

```
> ls
```

```
>> bin.file bin.file.uu
```

The binary file is now ready for use.

## ::SA 285::

bxvi Main Course Objectives:

- Install a server system
- Change system run levels
- Understand the configuration of devices
- Maintain disks and file systems
- Backup and restore file systems
- Configure the NFS environment for sharing and accessing remote file resources
- Add diskless client systems
- Configure terminals
- Configure the NIS+ environment to facilitate sharing system information consistently and automatically in a local area network

bxvii There are 453 user commands, 400 System Administration Commands

[Make a Windex file] - Windows Index file

```
> catman -w& (allows > man -k boot (keyword search))
```

```
> rm -rF / (This command would wipe out the entire drive)
```

b1.3 Multi-user - enables more than one user to access the same system resources

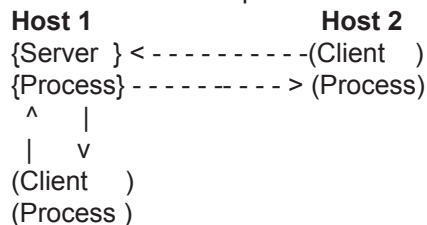
Multitasking - The kernel is able to run multiple processes simultaneously

Distributed processing - Enables the use of resources across the network

b1.4 **[The Client-Server Model for Networked Workstations]**

A server is a host or process that provides services to other machines on the network.

A client is a host or process that makes use of services made available by other machines on the network.



### **[Types of Servers]**

- NFS server (file system)
- Name services server
- Application server
- Mail server
- Fax server
- Operating system server
- Print server
- (Tape) Backup server

b1.5 **[NFS Distributed File Systems]**

- Common read-only file systems can be shared over the network (or between systems)
- Home directories can be made available over the network so that users can access their own files from any workstation
- Writeable file systems can be shared over the network with specific access rights given.
- Workstations with no local disks can boot up and mount all needed file systems and swap space from a remote system over the network.
- Workstations can even boot from a remote system with a different software release or kernel architecture.
- Workstations with minimum disk space can use NFS for their root and swap partitions and get the rest of their executables and data files from other systems.

### **[NIS+ Information Services]**

•NIS+ enables system administration functions such as adding users to be centralized on a server.

b1.7 **[Relationship between various Sun systems and their architectures]**

System Name	CPU	Kernel Architecture	Application Architecture
Sun-4/1xx series	4100	Sun-4	Sun-4
Sun-4/2xx series	4200	Sun-4	Sun-4
Sun-4/3xx series	4300	Sun-4	Sun-4
Sun-4/4xx series	4400	Sun-4	Sun-4
SPARCstation 1	4/60	Sun-4c	Sun-4
SPARCstation 1+	4/65	Sun-4c	Sun-4
SPARCstation SLC	4/20	Sun-4c	Sun-4
SPARCstation IPC	4/40	Sun-4c	Sun-4
SPARCstation 2	4/75	Sun-4c	Sun-4
SPARCstation ELC	4/25	Sun-4c	Sun-4
SPARCstation IPX	4/50	Sun-4c	Sun-4
SPARCstation 600MP	600MP	Sun-4m	Sun-4
SPARCstation 10	SS10	Sun-4m	Sun-4
SPARCclassic	4/15	Sun-4m	Sun-4
SPARCstation LX	4/30	Sun-4m	Sun-4
SPARCstation Voyager	Voyager	Sun-4m	Sun-4
SPARCstation 5	SS5	Sun-4m	Sun-4
SPARCstation 20	SS20	Sun-4m	Sun-4
UltraSPARCstation	UltraSPARC	Sun-4u	Sun-4
SPARCsystem 1000	SS1000	Sun-4d	Sun-4
SPARCcenter 2000	SC2000	Sun-4d	Sun-4
SPARCengine 1E	4/E	Sun-4e	Sun-4

(The Ultra has a 64bit address size, the old code name was Dragon. E also means 3<sup>rd</sup> party engine)

The boot server configuration is referred to as the operating system server (OS server).

An OS server is a standalone configuration with added OS support for diskless clients.

Additional kernel architectures can be installed to support Sun workstations as diskless clients.

b2.8 **[Installation Process Overview]**

Install process

-sysidtool - A suite of programs used to configure the identity of the new system

-installtool - A utility used to install a Solaris 2.x standalone, dataless or server system. The upgrade option can be used to upgrade a system from an older release of Solaris 2.x provided the disks do not need to be repartitioned.

-Admintool - A graphical user interface used to perform post-installation tasks, such as adding users.

b2.9 **[Installation Information Worksheet]**

Host name

IP address

Name service (NIS,NIS+,none)

Domain name

Name server host name

Name server IP address

Connected to network

Subnet

Subnet Mask

Location and Time Zone

OS server - number of diskless clients

OS server - architecture type

Configuration cluster

System disk

File systems - root / 0, 1 swap, 2, 3, 4, 5, 6, 7

Root password

b2.28 **[OS Server and Diskless client setup]**

If you choose OS server, a window will pop up asking for information about the diskless client support.

If you choose Dataless client, a window is displayed. You are asked to specify the server host information for NFS mounting the /usr and /usr/kvm file systems.

- b2.29 In the “Select Platforms” window, If a sun-4m server is supporting a sun-4c diskless client, check both architecture types.
- b2.30 “Allocate Client Services” box  
Service = Both  
Clients # = 1 x 15  
Swap = 32
- b2.33 “Disks”  
If the drive doesn’t show up on the left, the system does not recognize it.
- b2.35 “Preserve Data”  
If you want to preserve data on the drive, click preserve here, and select the partitions you want to preserve.
- b2.36 If you want to preserve / , /usr, /var ; you must rename them. But you can preserve /export/home or other partitions.
- b2.38 “Automatically Layout File Systems” (Good idea to choose this and manually adjust later).  
The server installation enables you to select /export, /export/root, /export/swap, and /export/exec in addition to those for the standalone system.
- b2.40 Click on “Customize”, and adjust the size of each partition. The “disk icon” in the upper left assigns small clusters.
- b2.42 **[Mount Remote File Systems]**  
Click on “Remote Mounts?” (NFS Server)  
Enter the necessary information for NFS mounts in this window. Server name, IP address, and so on. You can test the mount before the install.  
Ex:  
Server: river1  
IP address: 155.60.20.4  
Remote file system: /usr/share/man  
Click on “Add”  
Local mount point: /usr/share/man  
(An NFS server must be sharing resources for mounting).
- b2.47 **[Exploring Post-Installation Files]**  
The /var/sadm directory contains subdirectories with summary information regarding the previous installation.  
•/var/sadm/install - contents file, a listing of every object installed on the system.  
•/var/sadm/install\_data - install\_log file, a log of the installation process.  
•/var/sadm/pkg - every package installed on the system, which includes detailed package information.  
•/var/sadm/softinfo - inst\_release file, Solaris version number.
- b3.3 **[Admintool]**  
GUI interface for: System Databases, printers, software, serial ports  
Quickly: Add users and groups, install and remove printers and software, enable/disable serial ports.  
Software must be available locally, either on a mounted CD-Rom or on a hard disk. Admintool extracts the specified software from the local source and installs it on the local system’s disk. Admintool cannot install software on remote systems.
- b3.6 **[Admintool: Add User]**  
Secondary Groups: 14 (This is a Sys Admin Group)
- b3.9 **[Solstice AdminSuite]** = Provides tools to manage system information in a network environment.  
•Manage local or remote hosts  
•Modify local or remote hosts administration databases.  
•Centralize system data using NIS or NIS+  
The following packages must be installed prior to installing Solstice AdminSuite. To install Solstice you must be a non-root user with membership in the sysadmin (GID of 14) group.  
•SUNWadmc  
•SUNWadmfw  
•SUNWmfrun  
•SUNWsadml (This is the Solstice launcher package)  
•SUNWlibCf  
•SUNWfns  
•SUNfnspr
- b3.10 **[To install Solstice]**  
> cd /cdrom/adminsuite\_2\_1  
> ./admin\_install  
> select “1” (Install AdminSuite 2)  
> select “Station Manager”  
> select “1” ( /export/opt)

> select "2" (Managed clients will mount via /etc/vfstab).  
 > select "1" (Sparc)  
 > type your host name  
 > select "1" (Start Installation)  
 > /usr/bin/solstice & (to run the program)

b3.13 **[Remove or add Software with Admintool]**

> admintool &  
 > click on Browse, Software  
 > select an item, click on "Show Details"  
 > click on Edit, Delete (To remove; Add to add-will need the Solaris CD-Rom)  
 (SUNWadumo needs SUNsound)

b4.3 **[::Package Commands]**

All bundled and unbundled software is distributed as packages on a Solaris 2.x system. Packages contain:

- Files describing the package
- Files describing the relationship to the target system (for example, disk space required)
- The actual files to be installed
- Scripts that should be run before and after installation (or during removal)

After installation, software packages are added to the system using the package administration commands or Admintool.

b4.4 **::pkginfo => pkginfo [ -d *device* | *pathname* ] -l *pkg\_name***

Options

- pkg\_category - The package category can be an application or a system
- pkg\_name - The software package name; if it begins with SUNW, it is a Sun product
- description - This column provides a brief description of the software product

Ex: To display a listing from the CD-Rom

> pkginfo -d /cdrom/cdrom0/s0/Solaris\_2\_5\_1 | more

Ex: To display detailed information about a package, including the package size (in disk blocks)

> pkginfo -d /cdrom/cdrom0/s0/Solaris\_2\_5\_1 -l SUNWaudio

(Note: the last line will indicate the size of the package)

b4.6 **::pkgrm => pkgrm *package\_name***

The pkgrm command uses the /var/sadm/install/contents file to determine where files are located, and pkgrm updates this file after a package is removed. If a file is shared by two or more packages, it is only removed when the last of those packages is removed.

b4.7 **::pkgadd => pkgadd [ -d *device* | *pathname* ] *pkg\_name***

Ex: pkgadd -d /cdrom/cdrom0/s0/Solaris\_2\_5\_1 SUNWaudio

b4.8 **::pkgchk - This command verifies that the attributes and contents of packages path names are correct by comparing them to their values as specified in the system log file. This command can check an entire package or just a particular path name if used with the -p option. No output is given if no disparities are detected.**

=> pkgchk [-p path1 [path2?]]

b4.9 **[::/var/sadm/install/contents]- The /var/sadm/install/contents file, is a log file of all packages installed on the system. This makes it possible to identify the package(s) that contain a particular file (or command) or related files.**

[Search the /var/sadm/install/contents log file]

> grep /usr/sbin/pkgadd /var/sadm/install/contents

**[To identify the directory location of a command] > grep pkgadd /var/sadm/install/contents**

b4.10 A package can be copied from the installation CD-Rom without installing it so that it can be stored on a system. For example, a server can copy software packages for storage, and then a system without CD-Rom player can mount the software packages from the server for installation. Software packages can also be installed remotely using the software management tool.

**[Spool packages on the server into the /var/spool/pkg directory]**

> pkgadd -d /cdrom/cdrom0/s0/Solaris\_2\_5\_1 -s SUNWaudio

-s copies the package into the /var/spool/pkg/ directory by default

(you can specify a different directory as an argument to the -s option)

Ex: pkgadd -d /cdrom/cdrom0/s0/Solaris\_2\_5\_1 -s /export/pkg SUNWaudio

Packages can be removed from a spooling directory with the -s option.

> pkgrm -s /export/pkg SUNWaudio

**[Summary table of files and directories used to perform package administration.]**

<b>File or Directory</b>	<b>Description</b>
/var/sadm	contains system log and administration files

`/opt/pkgname` Preferred location for the installation of unbundled *pkgname* package  
`/opt/pkgname/bin` Preferred location for the executable files of unbundled *pkgname* package  
or `/opt/bin`  
`/var/opt/pkgname` Preferred location for log files of unbundled *pkgname* package  
or `/etc/opt/pkgname`  
`/var/sadm/install/contents` Package Map of the entire system

b5.5 **[::Patches]**

Patches are assigned numbers, and are packages in a directory named with the patch number, such as "101945-34." The patch number is 101945, and 34 is its revision. The directory will contain a readme file that describes the patch as well as the patch files. It will also contain install and backout scripts to add or remove the patch to the system software.

[show installed patches] > showrev -p

**[/var/sadm/patch]** - This is created as soon as the first patch is installed, this is where the patches are kept.

**[Get the latest Monthly report]**

> ftp sunsolve1.sun.com

> cd /pub/patches

> ls

> get Solaris 2.5.PatchReport

> quit

(Note: There is a charge for some patches)

b5.8 **[Install a patch]** - (In this example it is a patch for the Mail utility)

> cd /var/sadm/patch

> showrev -p |grep 102042 (To see if it already exists)

> cd /tmp

> ftp sunsolve1.sun.com (use anonymous ftp)

> cd /pub/patches

> ls \*

> bin

> get 102972-01.tar.Z (don't need to add the Z)

> quit

> uncompress 102972-01.tar.Z

> tar -xvf 102972-01.tar

> cat readme |more

> ./installpatch .

(You will usually need to reboot)

b5.11 **[Install a patch from CD]**

> cd /cdrom/cdrom0/s0/Patches

> ls

> cd /cdrom/cdrom0/s0/Patches/OW (or whatever patch you are looking for)

> cat readme |more

b5.12 **[Back out of a patch]** - If the system exhibits problems after a patch upgrade.

> cd /var/sadm/patch

> ls (Look for the backout script)

> ./backoutpatch 102972-01 or ./102972-01/backoutpatch

> showrev -p

b6.3 **[::Boot PROM]**

Prom Levels

The open boot programmable read-only memory (OBP PROM) is used on all Sun Sbus type systems for booting systems.

*Three boot prom levels are covered in this module:*

<b>System Type</b>	<b>Architecture</b>	<b>OBP PROM</b>
IPX,SS1,SS1+,SLC	4c	1.x
IPX,ELC,SS2	4c	2.x
SS5,SS10,SS20,Classic, LX	4m	2.x
Ultra	4u	3.x

**[Check PROM level]**

ok > banner

>> ROM rev. 2.15

b6.4 **[::Device Terminology]**

art/sa285\_06-04.gif

**::Bus** = A pathway between hardware devices upon which a particular communication language is used. All current Sun systems use an internal bus called the system bus (sbus), while previous generations used a bus called versa modula europa (VME). The peripheral device bus that connects disks to the Sbus is a SCSI.

**::Interface Controller (Host Adapter)** = An interface device translates instructions between an Sbus and a device bus. For example, SCSI host adapters connect disks with embedded device controllers to the Sbus.

**::Device Controller** = A device controller is circuitry that controls the device. SCSI disks have an embedded device controller.

art/sa285\_06-05.gif

**::Device Driver** = Programs will not communicate with a device or their controllers directly. Instead, programs will communicate with a device by way of a device driver. A device driver is a routine within the kernel of the operating system that communicates with the device or its controller.

**::Device Files** = Device files are special files that contain information about the device drivers. A device file provides a pointer into the kernel for programs. It also enables the users to access devices as if they were a file.

b6.6 **[::PROM Monitor]**

The OBP PROM monitor is a firmware utility for initializing the system prior to boot. It enables greater functionality than previous type PROMs, including the ability to specify booting paths to many more device addresses. The PROM monitor also provides an interface to the physical devices attached to the system through buses:

- The Sbus is the internal bus on the motherboard.

- SCSI is the peripheral bus.

**[::Interface Controllers (Host Adapters)]**

SCSI

SCSI devices are connected to a SCSI host adapter that is used with both Sbus and VME-based systems. Only the SCSI interface is used to connect CD-Rom drives.

Intelligent Peripheral Interface (IPI)

IPI disk drives are connected to IPI controllers that are used only in systems with a VME bus.

(This module focuses on the SCSI bus.)

b6.8 **[System physical Device Names]**

SPARCstation 10 and newer systems

(I/O memory)

**/iommu@f**, (Management Unit) **/sbus@f**, (1st Sbus controller) **/espdma@f**, (1st SCSI DMA controller) **/esp@f** (1st SCSI host adaptor) **/sd@3,0:a** (**sd@3** SCSI target address, **0** SCSI LUN (Logical Unit Number), **a** Partition or slice {a=0,b=1,c=2})

I/O MMU = SPARCstation 10 and newer systems have a separate I/O memory management unit (iommu) on the CPU board.

Sbus = System bus controller.

DMA = A direct memory access controller is a specialized device that enables data to be transferred directly between a device such as a disk or Ethernet port and memory without going through the CPU.

SCSI host adapter = The built-in SCSI host adapter connects to the SCSI DMA controller.

**sd@#,0:a** = Identifies the SCSI device at target addresses, disk number (LUN), and slice.

SPARCstation 2 and earlier Sbus systems

Ex: The following example describes the SCSI disk at target address 3 connected to the first SCSI host adapter.

**/sbus@1**, (1st Sbus controller) **/esp@0**, (1st SCSI host adapter) **/sd@3**, (SCSI target address) **0** (SCSI LUN) **:a** (Partition or slice)

b6.12 **[OBP PROM 1.x and 2.x preset naming conventions]- {{See A18.5}}**

Target	Device	Description
3	disk	Boot disk
0	disk3	Disk
1	disk1	Disk
2	disk2	Disk
3	disk0	Disk
4	tape0	Tape Drive
5	tape1	Tape Drive
6	cdrom0	CD-Rom
7	adapter	SCSI controller

**[OBP PROM 2.x preset naming conventions] - (Implemented around 1994)**

Target	Device	Description
3	disk	Boot disk
0	disk0	Disk

1	disk1	Disk
2	disk2	Disk
3	disk3	Disk
4	tape0	Tape Drive
5	tape1	Tape Drive
6	cdrom0	CD-Rom
7	adapter	SCSI controller

**[OBP PROM 3.x preset naming conventions] - (Ultras)**

Target	Device	Description
0	disk	Boot disk
0	disk0	Disk
1	disk1	Disk
2	disk2	Disk
3	disk3	Disk
4	tape0	Tape Drive
5	tape1	Tape Drive
6	cdrom0	CD-Rom
7	adapter	SCSI controller

**b6.15 [Identify boot devices]**

ok > devalias

ok > boot disk (for example)

**b6.17 {{[See A18.9]} [Change PROM settings for boot disk]}**

ok > printenv (to see which device the system is booting from)

ok > setenv boot-device disk2

ok > printenv boot-device (To verify the change)

ok > reset

**b6.18 ::boot => ok boot [device\_name] [options]**

a - Performs an interactive boot that prompts for root and swap devices and several important system files.

r - Performs a reconfiguration boot where the system probes all attached devices and creates entries in the / devices and /dev directories and updates the files /etc/path\_to\_inst

s - brings the system to run level S.

v - Displays detailed startup messages.

**Boot Examples**

[boot Sun platforms] - ok > boot

[boot from a specific drive] - ok > boot disk2

[boot a system as a diskless client] -ok > boot net

[reconfigure peripherals] - ok > boot -r

[boot to single user mode] - ok > boot -s

[boot to single user mode from CD-Rom] - ok > boot cdrom -s

**[::Prevent Stop-A]**

> setenv security-mode true (Prevents Stop-A commands)

**b6.21 ::sync =** If a system hangs and you cannot log in remotely for any recovery attempts, you may have no choice but to interrupt the system. Interrupting the system stops the processor, it does not clear memory. And it does not synchronize the file systems (write changes to disk with the sync command) or provide any warning messages.

ok > sync (shutdown and init 6 automatically issue a sync command)

ok > go

**b7.3 [Boot Terminology]**

**[::Run Levels]**

Run Level      Function

0      PROM monitor level

1      Administrative state (single-user state with some file systems mounted and user logins disabled. Allows rlogin or telnet)

2      Multi-user level (with no resources shared - no NFS)

3      Multi-user level {default} (with resources shared - NFS enabled)

4      Not currently used

5      Halt (and power off Sun-4m and Sun-4u architectures)

6      Reboot to default level 3

S,s      Single-user state with some file systems mounted and user logins disabled

**[::Run Control scripts]**

The Solaris 2.x system provides run control (rc) scripts that change the run level. These changes included

functions such as:

- Checking and mounting file systems (fsck)
- Starting or stopping processes such as the print daemon, the sendmail daemon, and NFS client-server daemons.
- Performing housekeeping tasks

The run control scripts are used to build a specific run level.

b7.4 **[Single-user]** = Single-user state means the virtual console terminal is assigned to the system console. It is often referred to as the superuser or maintenance level. You must know the root password to get to single-user mode.

Only minimal processes are running, and no users can log in.

[Multi-user] = All defined terminal and daemon processes are running.

b7.5 **[System Boot Phases]**

Boot PROM Phase	PROM runs self-test diagnostics. PROM loads the bootblock (bootblk) program.
Boot Program Phase	The bootblock program loads the boot (ufsboot) program. The boot (ufsboot) program loads the kernel.
Kernel Initialization Phase	The kernel initializes itself and starts the init process.
The /sbin/init Phase	The init process starts the run control scripts

#### Boot PROM Phase

The boot PROM performs the following steps during the first part of the boot sequence.

##### **1.It displays the system identification banner.**

The model type, keyboard type, PROM revision number, amount of installed RAM, PROM serial number, Ethernet address, and host ID are displayed.

##### **2.It runs self-test diagnostics.**

The boot PROM runs a self-test routine to verify the system's hardware and memory. It then begins its boot sequence upon successful completion of the self-test diagnostics.

##### **3.It finds the boot program from the default boot device programmed into the PROM.**

The boot PROM reads a system's primary boot program called bootblk (located at sectors 1-15) that contains a ufs file-system reader.

##### **4.It loads the boot program.**

The file-system reader opens the boot device, finds the secondary boot program called /platform/'uname -i' /ufsboot, and loads it into memory.

#### Boot Program Phase

After loading the /platform/'uname -i' /ufsboot program, the boot PROM loads the platform specific kernel (/platform/'arch -k' /kernel/unix) and the generic kernel (/kernel/genunix).

#### Kernel Initialization Phase

The kernel begins loading modules using the /platform/'uname -i' /ufsboot program to read the files as soon as it initializes itself.

When the kernel has read in the modules needed to mount the root partition, it unmaps the /platform/'uname -i' /ufsboot program from memory and continues initializing the system using its own resources.

#### The /sbin/init Phase

The kernel creates a user process and starts the /sbin/init program. The /sbin/init program starts processes by using information in the /etc/inittab file.

The /sbin/init program has two main roles:

- It creates processes, which has the effect of bringing the system up to the default run level.
- It controls transitions between run states by re-reading the /etc/inittab file.

The init process executes an rc script, or scripts, that executes a series of other scripts. These scripts (/sbin/rc\*) check file systems and mount file systems, start various processes, and perform housekeeping tasks.

b7.9 **[:/etc/inittab]**

The /etc/inittab file contains four fields

id - One to four characters used to identify an entry (Must be unique)

rstate - Defines the run level to be processed

action - Keywords tells init how to treat the process

process - Defines the command (or script) to execute

Ex: > s3(id):3(rstate):wait(action):/sbin/rc3(process) > /dev/console 2>&1 (standard error) < /dev/console

#### **[Action keywords]**

*initdefault* - Identifies the default run level, which is 3, by default, on Sun systems.

*respawn* - Starts the process and restarts it when it dies (Ex: login prompt)

*powerfail* - Starts the process when the init receives a power fail signal. (Read and go)

*powerwait* - (Read and hangon)

*sysinit* - Starts the process before trying to access the Console and waits for its completion before continuing.

*wait* - Starts the process and waits for it to finish before going on to the next entry for this run state.

(See the inittab man page for more)

Note: ? => label:3:flag:program name

Note: is:3:not default:when init starts, it looks here to determine the run level

Note: /sbin/rc3 = looks in /etc/rc3.d (run level) and looks for S (start) or K (Stop)

b7.10 The inittab entries tell the init process what processes to create for each run level and what actions to perform.

The /etc/inittab file defines three main items for the /sbin/init process:

- The systems default run level

- What actions to be taken when the system enters a new run level

- What processes to start, monitor, and restart if they die

b7.11 **[Diagram of the startup sequence - System Boot Cycle]**

The init process and the /etc/inittab file

init process----->

/etc/inittab

set initdefault = level 3

Run entries with sysinit-----> /sbin/autopush

in action field

|  
/sbin/rcS

|  
|  
|

Run entries with 3 in -----> /sbin/rc2

rstate field

|  
/sbin/rc3

|  
/usr/lib/saf/sac

|  
/usr/lib/saf/ttymon

|

System startup

1.The init process reads the inittab file.

2.The init process scans for the default run level by reading the initdefault entry.

3.The init process executes the commands or scripts for entries that have sysinit in the action field., such as /sbin/autopush. (autopush configures a list of kernel modules to be made available when a device is opened.

4. The init process executes the scripts for any scripts for any entry that has a 3 in the rstate field, such at the /sbin/rc, /sbin/rc3,/usr/lib/saf/sac, and /usr/lib/saf/ttymon executables.

b7.12 **[::Run Control Scripts]**

The following table summarizes the purpose of each script.

**Script Name    Purpose**

/sbin/rc0            This script is run to bring the system down to the PROM level. Systems services and all running processes are stopped. All the systems are unmounted.

/sbin/rc1            Historically, this script was used to bring the system down to run level 1. To get to run level 1 now, the system runs the shutdown command as noted in the /etc/inittab file.

/sbin/rc2            This script is run to bring the system to run level 2 - multi-user state. All file system are mounted, and all network services are configures except for sharing file systems.

/sbin/rc3            This script is run to bring the system to run level 3- multi-user state with shared file resources. The nfs server processes are started.

/sbin/rc5            This script is run to halt the machine and perform a power off for all SUn-4m and Sun-4u architectures. All system processes and services are stopped, and all file systems are unmounted.

/sbin/rc6            This script is run to halt and reboot the system. The /etc/rc0.d/K\* scripts are executed. All system services and processes are stopped, and all file systems are mounted. Then the initdefault entry in /etc/inittab is executed.

/sbin/rcS            This script is used to set up minimal network connectivity for diskless and dataless client support, and check and mount root (/) and /usr file system during all run level changes. Device entries are created.

b7.14 **[::Run Control Cycle]**

[The S\* and K\* files]

The scripts for starting and stopping system services are located in the /etc/rc#.d directories. Files that start with S

are used to start processes. Files that start with K are used to kill processes.

### The Start Files

In the /etc/rc3.d directory, for example, is a script for controlling NFS processes. (S15nfs.server)

### The Kill Files

Changing to run level 0 means the system is shutting down, so all processes must be killed. This means /etc/rc0.d contains only K\* files.

Because the scripts are run in alphanumeric order, the K57sendmail is run before the K66nfs.server file.

Note: If you want to test a startup file and need to disable another one, rename the file to be disabled by using a lowercase "s" or "k" at the beginning of the name.

#### b7.15 **[::/etc/rc\*.d]**

art/sa285\_07-15.gif

The rc scripts are contained in the /sbin directory. They direct the system to read files in the related run level directories located in the /etc directory. For example, the /sbin/rc2 script would read files located in the /etc/rc2.d. Both types of files are run in alphanumeric order.

Note: For System V compatibility, there are rc scripts in /etc that are symbolically linked back to /sbin. For example /etc/rc2 is a symbolic link to /sbin/rc2.

#### b7.16 **[::/etc/init.d]**

art/sa285\_07-16.gif

The start and kill files in the /etc/rc#.d directories that are used to start up or kill specific processes are hard linked to scripts in the /etc/init.d directory. This is done to facilitate booting.

An important advantage to system administrators of having the actual run control files in the /etc/init.d directory is that you can start and stop individual services or processes from a single location without changing run levels.

For example, you can stop and then start the lp print service using the following syntax:

```
> /etc/init.d/lp stop
```

```
> /etc/init.d/lp start
```

#### b7.17 **[Start and Kill Script for lp]**

```
> cat lp
```

```
(lp is a script, I have not included it)
```

```
state=$1
```

The \$1 is a command-line variable and represents the first argument. The script calls in \$1 and sets it to stat. The case statement then compares this variable to 'start' or 'stop'. If the variable matches 'start', then the command /usr/lib/lpsched is executed. If the variable matches 'stop', then the command /usr/lib/upshut is executed.

#### b7.18 **[add Run Control Files, How to]**

Follow these steps if you need to add startup or shutdown files for additional system services.

1. Read the instructions in the /etc/init.d/README file for an example of adding a startup file.

2. Assign an appropriate sequence number for the startup file in the /etc/rc\*.d directory.

- Do not allow a conflict with an existing sequence number.

• Ensure the service you are starting comes after a service that it needs to start successfully.

3. Place the startup file for the relational database in the /etc/init.d directory.

```
> vi /etc/init.d/rdbms
```

4. Create a link in the appropriate /etc/rc\*.d directory, based on the run level you want the service to be started in.

```
> cd /etc/rc3.d
```

```
> ln /etc/init.d/rdbms S22rdbms
```

```
> cd /etc/rc0.d
```

```
> ln /etc/init.d/rdbms K99rdbms
```

5. This process will now automatically startup during boot and shutdown from init.0. In addition, you can then start and stop this process from the command line.

```
> /etc/init.d/rdbms stop
```

```
> /etc/init.d/rdbms start
```

#### b7.19 **[modify Run Control Files, How to]**

Add the -Y option to the rpc.nisd (NIS+) daemon to provide network information service (NIS) compatibility.

```
> cp /etc/init.d/rpc /etc/init.d/rpc.orig
```

```
> vi /etc/init.d/rpc
```

```
> _ (remove the # symbol from the line that shows EMULP="-Y")
```

```
> wq!
```

```
> /etc/init.d/rpc stop
```

```
> /etc/init.d/rpc start
```

#### b8.3 **[Changing Run Levels]**

A *clean shutdown* means the operating system is shut down in an orderly fashion where all processes are

stopped, file systems are unmounted, and data in memory is copied back to disk.

- b8.7 **[:/etc/motd]** - Notify users of impending system shutdown
- Send messages to logged-in users with the wall command.
  - Send messages to a network of users with the rwall command.
  - Send electronic mail messages to affected users.
  - Use the /etc/motd (message-of-the-day) file to supply a message (to users who log in to the system) about scheduled downtime.
- b8.8 **::shutdown** - The /usr/sbin/shutdown command is used to change a system's run level. Its benefit is that it notifies users of the coming shutdown.
- => shutdown [-y] [-gseconds] [-irun\_level]
- y - Continues without intervention. Does not prompt for a "yes to shutdown"
- g - Enables you to specify a time (in seconds) before the system is shutdown. This overrides the 60-second default.
- i - Enables you to bring the system to a different run level than the default shutdown run level, S. Choices are run level 0, 1, 5, and 6.
- Ex: > shutdown -y -g0 (Same as > init 6)
- b8.11 **[/usr/sbin/halt]**  
This is similar to init 0, but it does not run the rc0 scripts. Not an orderly command. (Hide this command).  
**[/usr/sbin/poweroff]**  
This command halts Sun-4m and the Sun-4u and then powers them off. It is similar to init 5, but it does not run the rc0 scripts.  
**[/usr/sbin/reboot]**  
Performs a clean shutdown and brings the system to run level 3 by default. Similar to init 6, except that the rc0 scripts are not run.  
Boot options may be passed to the boot command using the -- delimiter when reboot is run.  
[reconfigure during boot up] > reboot -- -r  
[boot to single user level] > reboot -- -s
- b9.3 **[Autoconfiguration]**  
The Solaris 2.x kernel consists of a small static core and a series of modules that are loaded on demand. A kernel module is a hardware device driver or software component that is used to perform a specific task within the system. An example of a loadable kernel module is a device driver that is loaded when the device is accessed.  
The kernel modules are loaded into memory.
- | <b>Kernel Modules</b>            | <b>Memory</b> |
|----------------------------------|---------------|
| /platform/'uname -i'/kernel/unix | Static Core   |
| /kernel/genunix                  |               |
| SUN                              | Driver Module |
| /platform/'uname -i'/kernel      | Streams       |
| /kernel /usr/kernel              | UFS module    |
|                                  | NFS module    |
- At boot time, the system does a self-test and checks for all devices that are attached to it. The kernel then configures itself dynamically, loading its modules into memory, as needed. After booting, a device driver is loaded when a service, such as the tape device, is accessed. This process is called auto-configuration because all kernel driver modules are loaded automatically when loaded.
- b9.4 **[:/kernel]**  
Each subdirectory of /kernel contains a particular type of module.
- ../drv Contains device drivers and pseudo device drivers
  - ../exec Contains modules used to run various executable files
  - ../fs Contains file-system modules (ufs,nfs, and proc)
  - ../misc Contains miscellaneous modules needed for virtual memory operation and inter-process communication
  - ../sched Contains scheduling classes and corresponding dispatch table modules
  - ../sys Contains loadable system calls such as those involved with semaphore operation
- b9.5 **[:/usr/kernel]**  
../drv; ../fs; ../misc; ../sched; ../strmod; ../sys  
The /usr/kernel directory is also used to store loadable kernel modules. This directory contains:
- Additional modules that are not needed to complete the boot process, such as the system accounting module.
  - Modules that can be shared across platforms, such as the audio driver.

All directories are used to load modules during the boot procedure.  
If a module is not loaded at boot time, it can be loaded when a service is requested.

#### b9.6 **[::/etc/system]**

The `/etc/system` file can be customized to change the kernel configuration process. This configuration file is read at boot time and by default, contains only comments.

This file can be customized in several ways:

- Use the `moddir` variable to modify the search path of the modules to be loaded at boot time.
- Use the `exclude` variable to exclude modules, even if referenced.
- Use the `rootdev` variable to determine an alternate root device.
- Use `set variable=value` to override a default kernel parameter.
- Use `forceload` (absolute path)

#### **[Customize the /etc/system file]**

```
> cp /etc/system /etc/system.orig
> vi /etc/system
> _ set pt_cnt=100 (pt_cnt defines the number of configurable Pseudo-ttys. The default is 48.)
> wq!
> init 6
```

#### b10.3 **[::Device Names]**

art/sa285\_10-04.gif

In the Solaris 2.x environment, devices are referenced in three different ways:

**[Logical Device Names]** - Names used by system administrators to reference devices. These names are symbolically linked to their corresponding physical device names. The logical names are located in the `/dev` directory and are created at the same time as the physical names.

Logical device names are files that are soft links to physical devices in the `/device` directory.

The main subdirectories in the /dev directory are:

<code>cua</code>	Dial-out modems
<code>dsk</code>	Block interfaces to disk devices
<code>fbs</code>	Frame buffers
<code>pts</code>	Pseudo terminals
<code>rdsk</code>	Raw or character interfaces to disk devices
<code>rmt</code>	Tape drives
<code>term</code>	Serial line devices

**[Physical device names]** - Names that represent the full device path name in the device information hierarchy (or tree). The physical names are located in the `/devices` directory where the entries are created during installation or subsequent device configuration. The device file provides a pointer into the kernel for programs. A physical device name represents the full device path name in the device node hierarchy (or drive tree). Physical device files are found in the `/devices` directory and correspond to the device names used at the PROM level. When the system is booted with the reconfiguration option (`boot -r`), `drvconfig` creates a device hierarchy (or tree structure) that represents all devices attached to the system.

The topmost object in the hierarchy is called the root node of the device tree. An intermediate object below the root node has a device driver associated with it and is called a leaf or bus nexus node.

The kernel identifies physical location of devices by associating a node with an address, `nodename@address`, which is called the physical device name, for example `sd@1`.

art/sa285\_10-08.gif

**[Device Instance names]** - The kernel's abbreviation names for every possible device on the system. `dmesg` displays instance names such as `sd0` and `sd1`.

An instance name is the kernel's abbreviation for the physical device name. Instance names are allocated when the kernel finds an instance of a device for the first time.

An instance disk device name is an abbreviation for the physical device name.

Examples of instance names include:

<code>sdn</code>	<code>s=SCSI, d=disk, n=logical disk number</code> such as <code>sd0</code> for the first SCSI disk drive.
<code>stn</code>	<code>s=SCSI, t=tape, n=logical tape number</code> such as <code>st4</code> for the first SCSI tape device.
<code>idn</code>	<code>i=IPI-2, d=disk, n=logical disk number</code> such as <code>id0</code> for the first IPI-2 disk device.

**[::/etc/path to inst]** - (Do not edit this, the kernel builds this file)

In the Solaris 2.x environment, the instance name is bound to the physical name by references in the `/etc/path_to_inst` file.

The device instance is the number on the right side of the file (the number is in bold in the displayed output of each device). The kernel uses these names to identify every possible device instance.

On the first bus, the instance number is the same as the target number. However, if an additional SCSI controller is added to the system, the instance numbers would increment sequentially, but the target numbers would remain

the same (0-7).

### **[cleanup /etc/path\_to\_inst]**

```
> rm /etc/path_to_inst*
> rm /dev/rmt/*
> cd /devices/iom*/sub*/esp*/esp*
> ls st??
> rm st*
> reboot -- -a
```

### **[Bus-Oriented Controllers]**

Sun systems use `.../c#t#d#s#` {.../controller number,Target number,Disk number,Slice(Partition)}

Note: There is only one disk at the target address for SCSI drives, so the disk number is always set to d0. The logical disk device name is used by system administrators (and users) when using disk and file-system-related commands.

### **[Direct Controllers]**

Disks connected to direct (non-bus-oriented) controllers such as Xylogics 451 or 7053, do not have a target number in their logical device name. (`/c#d#s#`)

#### **b10.12 ::dmesg**

This command identifies the devices connected to the system using the instance names and physical device names. From this output, it is possible to determine the logical disk names.

The `dmesg` command also displays system diagnostic message, the operation system revision number, physical memory size, and other information.

Since `dmesg` is in a buffer and can get overwritten, it helps to get a clean `dmesg` for reference purposes. To do this, halt the system, perform a reset, and then reboot. log in and then redirect the output of `dmesg` to a file.

```
> dmesg |more
```

```
>> The instance name starts with 'sd#'. The physical name is /sbus@1, f.. esp0 = interface and controller number (c0). target=address of device controller. Lun= LUN number (d0).
```

#### **b10.13 :format**

The `format` output displays both the logical and physical device names.

The above `format` example identifies one SCSI disk device (`sd@...`), connected to the SCSI host adapter (`esp@...`), which is connected to the SCSI DMA controller (`espdma@...`), which is connected to the SBUS interface (`sbus@1...`).

IPI and Xylogics disk devices are prefaced with `ip` and `xd`, respectively.

(Note: C]-d to exit `format`)

#### **b10.15 ::prtconf**

The `prtconf` command displays a system's configuration information, including memory and peripheral configurations.

Note the use of device instance names to distinguish the possible devices that may be connected to the same interface. For example, the devices connected to the SCSI bus (`sbus`) are listed immediately beneath the `sbus` device listing.

The disk and tape device instance names correspond to possible address locations on the SCSI host adapter, which are called target addresses. Assigning target address locations is how the system distinguishes different devices connected to the same interface.

The "driver not attached" message means that no device driver for that device is in use at the time the `prtconf` command is issued, or there is no driver for that device instance.

#### **b10.16 [::add new devices, How to]**

When new devices are added to the system, a reconfiguration boot is performed to recognize the new devices. This process creates a new device information tree and the directories `/devices` and `/dev`.

Ex: **[::add a new disk]**

```
> touch /reconfigure
```

```
(or) ok > boot -r
```

```
> init 0
```

```
- power down
```

```
(> reboot -- -r will also perform a reconfiguration boot. Device drivers can also be set up directly from the command line using ::add_drv which in turn runs ::drvconfig.)
```

```
- power on the system. After configuring the system to recognize a disk, setup the partitions.
```

#### **b10.17 [::rebuild path\_to\_inst, How to]**

If the system does not recognize the device, even though the logical name exists, you may have a corrupted `path_to_inst` file.

You can perform an interactive boot using `-ar` options. With these options, the boot program prompts you for the location and the name of all of the important system files it needs to boot and rebuilds the `path_to_inst` file.

```
> rm /etc/path_to_inst
> rm /etc/path_to_inst.old
> init 0
> boot -ar (on x86)
```

(If the /etc/path\_to\_inst does not exist, the system will rebuild it.)

### b11.3 **::format**

The basic tasks for repartitioning a disk are:

- Resize the disk partition
  - Relabel the disk with the new volume table of contents (VTOC)
- The partition sizes can be set up manually or selectively format the disk drive configuration file /etc/format.dat.
- [New Disks]**
- New disks require different treatment depending on the type of disk and how the disk was prepared by the distributor.
- New disks purchased from Sun are formatted and labeled with a predefined partition table from the /etc/format.dat file.
  - New disks that are supported by Sun can have several predefined partition tables stored in the /etc/format.dat file, may be pre-formatted, and have a disk label.
  - New disks that are not supported by Sun that do not have any predefined partition tables in the /etc/format.dat file.

### b11.8 **[Physical Disk Geometry]** - {see a19.4}

**::dmesg** - The dmesg command identifies the devices connected to the system using the instance names (names used by the kernel to configure the system) and physical device names. The dmesg command also displays system diagnostic messages, the operating system revision number, physical memory size, and other information. (Note: The number of heads equals the number of tracks per cylinder.)

### b11.9 **[::Partitions]**

The advantages to partitioning are that it:

- Functionally organizes data.
- Enables the superuser to develop backup strategies.
- Is reliable.

Disks can be divided into partitions to separate data types. The most common reason for partitioning a disk is to facilitate a backup strategy.

art/sa285\_11-10.gif

The partition table primarily defines the boundaries of the partition and the number of cylinders in a partition.

Partition boundaries must begin and end with entire cylinders.

Partitions are described by an offset (distance from cylinder 0) and a size. The offsets and sizes for a disk's partition are defined by a partition table.

### b11.12 **[Errors of a Partition's Size]**

This diagram illustrates what happens if you change two partition sizes without adjusting the partitions' starting cylinder numbers (or offset).

#### **[Wasted Disk Space]**

Decreasing the size of a partition leaves wasted space between it and the next partition.

art/sa285\_11-12a.gif

#### **[Overlapping Disk Space]**

Increasing the size of a partition creates overlapping partitions.

art/sa285\_11-12b.gif

### b11.13 **[Disk Label (VTOC)]**

A disk's label, also called the disk's volume table of contents (VTOC), contains:

- Partition tables* for the disk
- An optional *volume name* that identifies the disk device
- Optional *partition tags* that name the standard mount points for each of the partitions. Tags are of limited value as they are not used by the operating system and are limited in their definition.
- Optional *partition flags* that label whether each partition is writeable or mountable. Partition flags are of limited value.

The disk label occupies the first sector of a disk.

### b11.14 **[Where partitions are stored]**

art/sa285\_11-14.gif

All formatted disks have a partition table as part of their disk label. A set of predefined partition tables are stored in a file named /etc/format.dat that can be read in by the format utility.

When you select a disk within format, the actual partition table stored in the disk's VTOC is read into memory and listed as the current label.

You can also select a predefined label from /etc/format.dat to be read in as the current label by using the select command. This label may differ from the actual label on the disk.

The current label can be modified, and then it can be:

- Written to the disk VTOC using the label command under the partition or format menu.

Or

- Named using name, under the partition menu. To store the partition layout to /etc/format.dat, use the save command under the format utility's main menu.

#### b11.15 **::prtvtoc - Display the Disk Label -**

```
> prtvtoc
```

#### b11.16 **::format -**

When the format utility is first started, each available disk is described by its logical name, marketing name, physical parameters, and physical name.

The marketing name usually begins with *SUNnnnn*. In the above example, the marketing name is SUN0424.

You are prompted to choose a disk from the disks currently recognized by the system. To choose a disk, type the number corresponding to its description.

#### b11.19 **[::Partition Table]**

The name of the partition table is displayed in parentheses in the first line of the table.

**The column headers of the table have the following meanings:**

Part        The partition number

Tag        The partition tag, which is used to help identify the partition.

Flag       The partition flags are: wm - write/mountable; wu - write/unmountable

Cylinders   The range of cylinders occupied by the partition.

Size       The size of the partition in megabytes.

Blocks     The size of the partition in cylinder/track/sector notation.

If the disk had a volume name, it would be listed after the word Volume in the first line of the table.

#### b11.20 **[Format a partition into three disks]**

```
> format
```

```
> _ partition
```

```
> _ print
```

```
> _ 0 (To change the root partition)
```

- The id tag is supplied by the utility; it identifies the standard file-system names.

- Partitions can have either write/mountable (wm) or write/unmountable (wu) permissions.

- The starting cylinder is the offset from the beginning of the disk.

- Partition sizes can be expressed in blocks, cylinder groups, or megabytes.

```
> _ input the size (for Ex: 1gig)
```

```
> _ print (The current partition table is displayed)
```

```
> _ 1 (To change the swap partition)
```

```
>> _ Enter new starting cyl: 813
```

Note: The new starting cylinder is one greater than the ending cylinder for partition 0.

```
>> _ Enter partition size: 200mb
```

```
> _ print
```

```
> _ 7 (To change the home partition)
```

```
>> _ Enter new starting cyl.: 935
```

```
>> _ Enter partition size: 1101c
```

Note: You should zero out all partition sizes prior to assigning new slice sizes if working on a new disk.

```
> _ print
```

(Add up the Size column to verify the accuracy of your work.)

#### b11.28 **[::Label the disk]**

The following adds the new partition table definition to /etc/format.dat. This enables a customized partition to be reused.

```
partition > _ label
```

Name the current table. Names frequently use the disk manufacture information.

```
partition > _ name
```

```
> _ quit
```

```
format > _ save
```

```
[How to use]
```

```
format > _ partition
```

```
partition > _ select, 3
```

```
partition > _ label
```

```
partition > _ quit
```

format > \_ quit

b11.30 **[Modify a partition]**  
 > format  
 > \_ Select a disk by typing the number to its description.  
 format > \_ partition  
 partition > \_ modify (Select partitioning base.) - 0  
 The Free Hog partition is used as a disk space accumulator that expands and contracts as other partition sizes are changed. This functionally is similar to the installtool utility's unlocated space meter.  
 Press Return to accept partition 6 (the default) as the Free Hog partition. If partition 6 has no space on it, you must specify one that does.  
 > \_ R]  
 partition > \_ print  
 > \_ R]  
 > \_ Type the size of partition 0 (200 mb in this example) and press return.(Let the other partition sizes default to their current sizes.  
 You are not prompted to change the size of partition 6 because it has been designated as the Free Hog. It is decreased in size because partition 0 increased (200mb). The new partition table is displayed.  
 Verify that the new partition sizes are correct.  
 The format utility rounds the megabytes to the next whole cylinder. Using the modify option (and designating a Free Hog partition) automatically adjusts the starting cylinder boundaries of the other partitions so there are no "holes" between partition boundaries.

b11.35 **[write to the VTOC, How to]**  
 Once the partition sizes are changed, the modified disk label is written to the disk VTOC.  
 > \_ Okay to make this the current partition table [yes]? R]  
 > \_ Name the partition table and press return. Partition table names usually contain some reference to the disk name and size. (Ex: "c0t0d0.424")  
 > \_ yes  
 > \_ quit

b12.3 **[Solaris File System]**  
 The UNIX file system (ufs) is based on the Berkley fast file system. It is designed to optimize disk performance through the use of cylinder groups and contiguous data blocks.

**[Definition of a File System]**

To the user of the Solaris operating system, a file system is the collection of files and directories used to store and organize information.

To the operation system, a file system is a partition that has been formatted into file system (data) blocks and contains a structure of tables defining the locations of files and directories.

**[Raw and Block Partitions]**

A raw partition has a defined beginning and size but has no file system installed. A partition can be used as a raw device, such as swap. A block partition has a file system. Logical device names distinguish between the two by the directories /dev/rdisk and /dev/dsk.

b12.4 **[The Disk Label (VTOC)]**  
 The disk label contains the partition table for the disk and is referred to as the Volume Table of Contents (VTOC). A disk partition can contain a file system that the Solaris operating system interprets as an organization of directories and files.

The disk label is located in the first disk sector (512 byte blocks).

**[The Bootblock]**

The bootstrap program (bootblk) is in the next 15 disk sectors. Only the root file system has an active boot block, although space is allocated for a boot block at the beginning of each file system.

**[The Superblock]**

The file system is described by its superblock. The superblock is contained in the 16 disk sectors following the boot block. The superblock is a table of information about the file system including:

- The number of data blocks
- The number of cylinder groups
- The size of a data block and fragment
- A description of the hardware (derived from the label)
- The name of the mount point

**[Backup Superblocks]**

Because the superblock contains critical data, it is replicated in each cylinder group to protect against catastrophic loss. This is done when the file system is created.

b12.5 **[Solaris ::File System Structure]**

Label  
Boot block

Superblock

First cylinder group    Backup superblock  
                                  Cylinder group block  
                                  Inode table

Data blocks

Second cylinder group

b12.6 **[::Cylinder Groups]**

By dividing the partition into cylinder groups (the default is 16 cylinders per group), disk access is improved. The file system will constantly optimize the disk by placing file data in one cylinder group, thus reducing head travel. The file system will store files across several cylinder groups, if needed.

**[Cylinder Group blocks]**

The cylinder group block is a table that describes the cylinder group, including:

- The number of inodes.
- The number of data blocks in the cylinder group.
- The number of directories.
- Free blocks, free inodes, and free fragments in the cylinder group.
- The free block map.
- The used inode map.

**[Inode Table]**

The inode table contains the inodes for the cylinder group. An inode (from the term "index node") is the internal description of a file and the location of its data blocks. Each cylinder group contains a portion of the total number of inodes.

**[Data Blocks]**

A data block is the unit of storage for data in the Solaris 2.x file system. The data block is 8192 bytes in size by default.

art/sa285\_12-07.gif

b12.8 **::inode**

The inode contains the following information about a file:

- The type of file, the access modes
- The user owner's and group owner's ID numbers
- The size of the file
- The time the file was last accessed, modified, and the inode changed
- The number of data blocks used by, or allocated to, the file

The inode contains two types of pointers: direct pointers and indirect pointers.

**[Direct Pointers]**

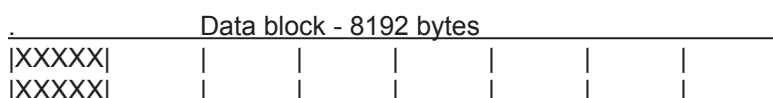
There are twelve direct pointers, which refer directly to data blocks. The twelve direct pointers can directly reference the data blocks of a file up to 96 Kbytes.

**[Indirect Pointers]**

- Single Indirect - A single indirect pointer refers to a file system block containing pointers to data blocks. This file system contains 2048 additional addresses of 8-Kbytes data blocks and points up to an additional 16 Mbytes of data.
- Double indirect - A double indirect pointer refers to a file system block containing single indirect pointers, each indirect pointer referring to a file system block containing the data block pointers. Double indirect pointers point up to an additional 32Gbytes of data.
- Triple indirect - A triple indirect pointer is not used in the Solaris 2.5 system.

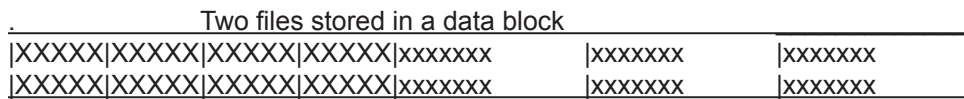
**[::Data Blocks and Block Fragments]**

The method for storing files that do not fill data blocks evenly is fragmentation of a data block. The data block is divided into 8 fragments of 1024 bytes each. Data from a file is written into a data block fragment, the minimum disk storage for any part of a file.

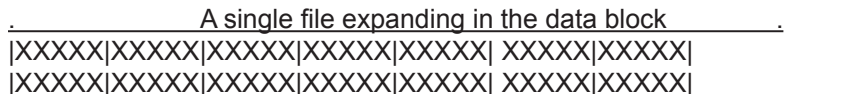


1024  
bytes  
**Fragment**

A file needing more disk space but not another file-system block is allotted fragments only. With fragments, it is possible for data from two files to be in the same file-system block.



If a file grows to require more space than is available in a block, it shares with a second file. The Solaris operating system moves all of the data of the file from that block into another block, thus assuring that all fragments are in the same file-system block. There will never be fragments for the same file in two file-system blocks.



#### b12.10 **::newfs**

##### **[create a file system, How to]**

After using the format utility to change a partition's size, the next step is to create a file system to add new data. The newfs command is a "friendly" front end to the mkfs command which actually creates the file system.

```
> newfs /dev/rdisk/c0t0d0s0
```

Caution: This command overwrites any data on an existing file system. It also creates a lost+found directory that is used by the file-system check and repair (fsck) utility.

Repeat the previous step for each partition that will be used to contain a file system.

Note: The newfs command will "reserve", by default, 10% of the disk space for file system maintenance. This may be adjusted using either newfs -m # option during creation of the file system or ::tunefs after the creation of the file system.

#### b13.3 **[Accessing File Systems]**

##### **[::etc/mnttab, ::etc/vfstab - Mounting Files]**

Mounting is the process by which separate file systems are attached to the file system hierarchy (file tree structure). File systems are attached at mount\_points, which are directories.

Mounting and unmounting create and remove entries in the **::etc/mnttab** (Mount Table) file. This file maintains a table of all mounted file systems.

Mounting and unmounting of file systems also occurs during system startup and shutdown based on the entries in the **::etc/vfstab** (Virtual File System Table) file.

It is not, however, uncommon for system administrators to encounter situations that require them to mount and unmount file systems from the command line. These situations include making backups, checking the file system for inconsistencies, and repartitioning.

The mount command will display all mounted file systems.

##### **[Local File Systems]**

Types of file systems stored on local physical media such as disk, CD-Rom, or diskette.

**ufs** The UNIX file system is the local disk file system based on the BSD fast file system.

**hsfs** The High Sierra CD-Rom file system.

**pcfs** The file system that supports read and write access to data on DOS diskettes.

#### b13.4 **::mount** - To identify file systems type > mount

*The fields displayed by mount:*

Field 1 The mount point (directory).

Field 2 Mount\_point on ufs or nfs file system.

Field 3 Names the partition containing that file system or special file system type.

Field 4 Specifies whether the file system is mounted read and write, or read only,

Field 5 The word "on" field 6.

Field 6 Specifies the date and time the file system was mounted.

(Note: This information is stored in the /etc/mnttab file.)

#### b13.5 **[::sbin/mount]**

The /sbin/mount command is used to attach either a local or remote file resource to the file system hierarchy.

```
=> mount [-F type] device_to_mount mountpoint
```

```
=> unmount mount_point
```

**-F type** Specify different file system types. This option is only required if not using the default ufs or nfs.  
**device\_to\_mount** Specify the file system resource.  
**Mount\_point** Specify the mount point (directory) on the local system (which must already exist).

**[::mount a file system]**

> mount /dev/dsk/c0t0d0s5 /opt

If the file resource is listed in the /etc/vfstab file, you can specify either device\_to\_mount or mount\_point on the command line.

> mount /usr/share/man

**[::unmount a file system]**

> unmount /opt

b13.7 **[::/etc/vfstab]** - (Must have seven fields for each entry)

The virtual file system table /etc/vfstab provides default entries for mounting file systems at boot time.

The format of the file is one record per line, seven fields per record, with a dash (-) indicating a null value for a field.

*Fields of the /etc/vfstab file:*

device to mount Identifies the logical device name of a file system.

device to fsck Identifies the logical (raw) device name of a local ufs file system.

mount point The mount point for the local file resource.

FS type Always ufs for local file resources.

fsck pass The ufs file system is checked if this field contains a value greater than zero; a dash (-) means no check. If the value is 1, each ufs file system is checked sequentially. If the value is greater than 1, then fsck checks multiple ufs file systems in parallel on different disks for maximum efficiency.

mount at boot Either yes or no indicating whether the file resource should be mounted when the system enters run level 2, or when the mountall command is issued.

Three exceptions are /, /usr, and /var (if /var is a separate file system). Both of these are mounted by the mount commands specified in the /etc/rcS.3/S30rootusr.sh script. Their mount at boot fields are set to no.

mount options A comma-separated list of mount options. For a local ufs file system, the options are ro or the default rw.

b13.9 **[::mountall - Mount All Local File Systems]**

*To mount all local file systems.*

> mountall -l

The -l option indicates local file systems. The mountall command reads the information in the /etc/vfstab file under the "mount at boot" field and mounts all file systems with yes and at boot.

*To unmount all local file systems.*

> unmountall -l

The unmountall command reads the information in the /etc/vfstab file under the "mount at boot" field and unmounts all file systems with yes and at boot.

b13.10 **[mount a new file system, How to]**

> mkdir /database

> mount /dev/dsk/c0t0d0s0 /database

> vi /etc/vfstab (If you want the file system to mount at boot).

b13.13 **[mount different types of file systems, How to]**

Different file system types have unique properties that affect the mechanics of mounting. The file system type must be specified when you mount it. The mount command has a -F option that is used to specify the type of file system being mounted.

If vold (Volume Manager) is not running, use the commands below to mount a diskette or CD-Rom.

> mkdir /psfs

> mount -F pcfs /dev/diskette /pcfs

> mount -F hsfs -o ro /dev/dsk/c0t6d0s0 /cdrom

b13.14 **[Mounting Different File Systems]**

File system type is determined in the following sequence:

From the -F option, if supplied

By matching the block device (if found) with a file system type in the /etc/vfstab file and using that type

By using the defaults specified in /etc/default/fs for local file systems and in /etc/dfs/fs types for distributed file systems

Assuming that /dev/dsk/c0t0d0s0 is a newly created file system not listed in the /etc/vfstab file, the following command should work because ufs is the default specified in the /etc/default/fs file.

> mount /dev/dsk/c0t0d0s0 /database

b13.15 **::swap => swap options argument**

Options

l List swap space  
a Add to swap  
d Delete from swap  
s Summary of swap space

**[Create a swap file]**

> mkfile 200m /database/swap

**[Add a swap file to swap space]**

> swap -a /database/swap

**[To delete a swap file while on line]** - (Deleting the swap file will stop swapping and empty the specified disk space.)

> swap -d /database/swap

b13.17 **[Add a swap file system]**

> vi /etc/vfstab

> \_ /dev/dsk/c0t2d0s1 - - swap - no -

> \_ wq!

> reboot

**[Add a swap file to the /etc/vfstab file]**

> vi /etc/vfstab

> \_ /database/swap - - swap - no -

b13.18 **[Restore Root password]**

> boot cdrom -s (Make sure the Solaris CD is in the drive)

> mount /dev/dsk/c0t3d0s0 /a

Do the necessary file repair. In this example delete root's password. Setting and exporting the TERM variable enables the vi editor to work.

> TERM=sun

> export TERM

> vi /a/etc/shadow

> cd /

> unmount /a

> reboot

b14.3 **[::fsck - File System Check Utility]**

The fsck utility uses known parameters and redundant information to check the disk for inconsistencies.

For example, each inode contains a link counter that is incremented each time a link to the inode is created and decremented each time a link is removed. The fsck program checks whether the value of each inode's link counter is equal to the number of directory entries containing the inode's number.

As another example, a data block should never be claimed by more than one inode except for small files using file system fragments. The fsck program checks whether any data blocks are claimed by more than one inode.

**[File System Corruption]**

The most commonly corrupted item in a file system is the summary information associated with the superblock.

This area is modified with every change to the file system's blocks or inodes.

There are various possible reasons why synchronization would not take place, including:

- Improper shutdown of the system
  - Power failure
  - Hardware failure
  - Physical move of the workstation
- >> fs\_clean = The file system was properly shutdown  
>> fs\_ok = Mounted

b14.4 **[Synchronizing Data]**

During normal file system I/O operations, the in-memory copy of data is not copied back to disk before the process continues to some other operation.

For added system performance, a copy of a file system's superblock is copied into memory during the mount process. Subsequent changes to the file system data are then reflected in the in-memory copy of the superblock.

The disk is automatically synchronized with memory data when the sync system call is run by:

- The fsflush daemon (every 30 seconds)
- The shutdown, reboot, halt, or init commands during a graceful system shutdown.

Proper system shutdown is important because the superblock information and data may be in transition if the system is suddenly aborted or its system power is shut off unexpectedly.

art/sa285\_14-04.gif

The fsck program is used to check and repair file system inconsistencies caused by improper system shutdowns.

#### b14.5 **::fsck**

The fsck utility checks for inconsistencies in all of these structures. Inconsistencies, checked in order, are as follows:

- Superblock summary information, including
  - File system size
  - Number of inodes
  - Free data block count
  - Free inode count
- Cylinder group block, including:
  - Free data blocks claimed by files
  - Correct free data block count
  - Correct free inode count
  - Inode information, such as:
    - Incorrect link counts in inodes, which indicate missing directory entries
    - Inconsistencies between inode size values and the number of data blocks actually referenced
    - Inodes allocated or unallocated
    - File system data block information, such as:
      - One data block belonging to several inodes
      - Data blocks marked as free but in use
      - Data blocks marked as used but free
    - Directory information, such as:
      - Illegal or unallocated inode numbers in directories

#### b14.5 **[When the fsck program is run]**

When the Solaris 2.x operating system is booted, the fsck program checks the state of each file system. If each file system state is determined clean, the fsck program exits without further checking.

The fsck program runs in two modes, noninteractive and interactive.

##### **[Noninteractive Mode]**

The fsck utility is run in the noninteractive (preen) mode by the system during a normal boot up. In this mode on a corrupted file system, it only makes changes (to the file system) that are known to be correctable without operator intervention. If an unexpected inconsistency is found, the fsck program terminates with a non-zero exit status, leaving the system in single-user mode. The system administrator must then run the fsck utility interactively.

##### **[Interactive Mode]**

When running interactively, the fsck program lists each problem it finds followed by a suggested corrective action. The system administrator must decide if the correction is to be made.

#### b14.6 **::fsck => fsck [-F fstype] [-V] [-y|Y|n|N] [-o fstype options] [-m] [special ...]**

##### Options

**F *fstype*** Specify the file system type on which to operate.

**V** Echo the complete command line, but do not run the command. This option can be used to verify and validate the command line.

**m** Check but do not repair. This option checks that the file system is suitable for mounting, and it returns the appropriate exit status.

**special** The special argument represents the block or character special device (for example, /dev/rdisk/c0t0d0s7) on which the file system resides. In general, the character special device should be used.

**y|Y** Assume a yes response to all questions

**n|N** Assume a no response to all questions

**o** The *fstype* options. The *fstype* options are interpreted by the file-system-specific part of the program. These options are specified in a comma-separated (with no intervening spaces) list of options or keyword-attribute pairs. Three of the common *fsck\_ufs* options are:

**b=n** Use block *n* as the superblock for the file system. Block 32 is always one of the alternate superblocks. The location of other superblocks may be determined by running *newfs -N*.

**p** The preen option is used during the automated startup procedure to make "safe" repairs.

**f** The force option is used to check a file system regardless of the state of its clean flag.

**Caution:** The fsck program should never be run on a busy file system. Because of its multipass processing, the fsck utility would report errors and possibly delete files as users were trying to read or write their data. Run the fsck utility in single-user mode or on unmounted file systems only. Be aware that you cannot unmount the root (/), (/var) and (/usr) file systems. Also, fsck, prior to backup.

#### b14.9 **[fsck Sample]**

The following is the output from the fsck program when no inconsistencies were discovered:

```
> fsck /dev/rdisk/c0t3d0s7
```

```
>> ** /dev/rdisk/c0t3d0s7
>> ** Last Mounted on /export/home
>> ** Phase 1 - Check Blocks and Sizes
>> ** Phase 2 - Check Pathnames
>> ** Phase 3 - Check Connectivity
>> ** Phase 4 - Check Reference Counts
>> ** Phase 5 - Check Cyl groups
>> 2 files, 9 used, 21606 free (14 frags, 2699 blocks, 0.1% fragmentation)
```

The first thing worth noting in this output is that the fsck program does its work in phases. This is one reason why it is critical to run this program on an inactive file system.

The last line lists the following values for the file system:

- The number of files used (2)
- The number of Kbytes used (9)
- The number of Kbytes free (21606)
- The breakdown of the free space into free blocks (2699) and free block fragments (14)
- The ratio of free block fragments to total Kbytes (0.1% fragmentation)

#### b14.10 **[fsck Examples]**

Without any arguments, the fsck program checks those entries in the /etc/vfstab file that have an entry in the device to fsck field and have a non-zero numeric entry in the fsck pass field.

```
> fsck
```

Check a specific partition using a device file.

```
> fsck /dev/rdisk/c0t0d0s7
```

Check the file system in a specific partition using a mount point.

```
> fsck /usr
```

Preen mode checks and fixes the file system in a noninteractive mode, and it exits immediately if there is a problem that needs intervention.

```
> fsck -o f,p /dev/rdisk/c0t0d0s6
```

#### b14.11 **[Troubleshooting with fsck]**

**[Alternative Superblocks]**

One possible corruption of the file system is that the superblock becomes corrupted. Since the information in the superblock is vital to checking the file system, the fsck utility can be told to use one of the backup superblocks with the -b option. When the -b option is used, the location of a backup superblock must be provided.

There is always a backup superblock at offset 32 of a file system. If this block is also corrupted there are more backup superblocks. To determine where the superblocks are located, the newfs command can be used.

The newfs command will display the locations of the superblocks while creating file systems. This information can be displayed without creating a file system by using the -N option with the newfs command.

**[Use an alternative superblock, How to]**

Ex: To use the backup superblock located at offset 32 of the file system.

```
> newfs -N /dev/rdisk/c0t0d0s7 (List the backup superblocks)
```

```
> fsck -o b=32 /dev/rdisk/c0t0d0s7 (Select one and run fsck)
```

#### b14.12 **[adjust a link counter, How to]**

If the fsck program discovers any inconsistencies during interactive operation, the program prompts the operator about what action to take.

In this example, the fsck program discovers that the value of a directory inode's link counter and the actual number of directory links are inconsistent.

Correcting a link counter is a safe action, so you can respond by typing yes, y or R].

**[salvage the free list, How to]**

If the fsck program discovers that the unallocated blocks count and the free block number listed in the superblock are inconsistent, it must be salvaged.

Salvaging the superblock is another safe action, so you can respond by typing yes, y or R].

**[reconnect an allocated but unreferenced file, How to]**

Ex: The fsck program discovers an inode that is allocated but is unreferenced (not linked in any directory). A yes response to the RECONNECT? question results in the inode being linked to the lost+found directory with its name being its inode number.

After concluding the interaction with the fsck utility (and mounting the file system), the system administrator can investigate the file.

Check the file type.

```
> file /export/home/lost+found/#788 or 788
```

```
>> /export/home/lost+found/#788:Data
```

If the file type is ASCII text, use cat or an editor to view the file.

> more /export/home/lost+found/#788

If the lost+found file is intact, copy it to its correct location.

### b15.3 **[::Backups]**

There are two types of backups:

- Full dumps - Dumps that backup the entire file system.

- Incremental dumps - Dumps that backup only those files that have changed since the last lower level dump.

#### **[Incremental Backups]**

The `ufsdump` command has 10 backup levels. Levels 1 through 9 are incremental backups. They backup those files that have changed since the most recent dump at a lower level. If no lower level backup exists, a full backup is performed. level 0 is always a full backup.

Incremental backup depends upon a file of the last dump dates. This file is `/etc/dumpdates`. From this source, it compares the date of the last incremental backup to the modification date of individual files. This determines whether a file requires backing up.

The `u` option of the `ufsdump` command creates or updates the `/etc/dumpdates` file. Only the various level dumps of file systems are listed, not directories or files.

> cat /etc/dumpdates

### b15.5 **[Sample Backup Strategies]**

<i>Mon</i>			<i>Tues</i>	<i>Wed</i>	<i>Thur</i>	<i>Fri</i>
------------	--	--	-------------	------------	-------------	------------

3	4	5	6	2		
---	---	---	---	---	--	--

3	4	5	6	2		
---	---	---	---	---	--	--

A full backup (level 0) is performed once each month.

The level 2 backup at the end of each week records the files that have changed since the level 0 backup at the beginning of the month.

During the week, a level 3, 4, 5, and 6 make daily backups.

<i>Mon</i>			<i>Tues</i>	<i>Wed</i>	<i>Thur</i>	<i>Fri</i>
------------	--	--	-------------	------------	-------------	------------

5	5	5	5	3		
---	---	---	---	---	--	--

5	5	5	5	3		
---	---	---	---	---	--	--

A full backup (level 0) is performed once each month.

The level 5 backups on Monday through Thursday record the files that have changed since the previous backup at level 4 or below.

The level 3 backup each Friday records the files that have changed since the level 0 backup at the beginning of the month.

### b15.8 **[::Tape Devices]**

All tape devices, regardless of their type, are referenced by their logical device names. The logical tape device names use the following format:

> /dev/rmt/*nlbn* = *n*(Logical tape number)*l*(Tape density h,m,l)*b*(BSD behavior)*n*(No rewind)

Tape device names are numbered from 0, independently of their type, and have several different parameters:

**Tape density** Five density values can be given: h(high), m(medium), l (low), c(compressed), u(ultra)

Densities vary depending on the tape drive. Check the manufacturer's documentation for the correct density. If not specified, the default, is high uncompressed. For a QIC-150 tape drive, all parameters cause the drive to have a capacity of 150 Mbytes.

**BSD behavior** When the letter *b* is specified, the drive assumes BSD behavior. This means that when reading past an end-of-file mark, it returns the first record of the next file, and when closing the no rewind device, it skips a tape space forward.

**No rewind** By including the letter *n* at the end of the device name, the tape is not rewound when the current tape operation is completed.

### b15.9 **::ufsdump**

The `ufsdump` command is used to back up a file system. The backup can be a full or incremental dump of the entire file system or individual files and directories.

=> `ufsdump options [arguments] files_to_dump`

#### Options

**0-9** Specify the dump level option. Level 0 is the lowest level (called the epoch level or a full dump), and level 9 is the highest level.

**u** Update the dump record (`/etc/dumpdates`) with the date and dump level of this file system backup.

**c** Set the blocking factor to 126 for all cartridge tape drives. This causes dump to write 63 Kbyte records instead of the default 32 Kbyte records.

**a** Create an on-line archive of the file names dumped to tape.

**f** Specify the device to which the files are written. It requires an argument that is the device name.

**files\_to\_dump** The `files_to_dump` can be the raw or block file system device name (`/dev/(r)disk/c0t0d0s0`), the

file system name (/export/home), or a file or directory name (/export/home/lister).

Note: The blocking factor is the number of tape blocks (512 bytes) to write before inserting an interblock gap.

**b15.10 [backup a file system, How to]**

Take down the system to single user mode, or at least unmount the file system you wish to backup. Make sure the system is off line before proceeding.

> fsck /export/home (or whatever the file system you want to backup)

Ex: Perform a full level 0 dump of the /export/home file system using a cartridge tape.

> ufsdump 0ufc /dev/rmt/0 /export/home

When the default tape device (/dev/rmt/0) is being used, it is not necessary to specify the unit with the f option. The following command performs the identical function as the one in the above example.

> ufsdump 0uc /export/home

**b15.12 [perform remote backups, How to]**

The ufsdump and ufsrestore commands can be used to perform a backup or restore on a remote tape device.

To perform a remote backup or restore, you must:

- Have root access privileges on the system with the tape device (entry in the *.rhosts* file)

- Specify the server:type\_device in the ufsdump or ufsrestore command line.

Ex: Perform a full level 0 dump of the /export/home file system to the remote tape device on mars using a cartridge tape.

> ufsdump 0uf mars:/dev/rmt/0 /export/home

**b15.13 ::ufsrestore** - The ufsrestore command extracts files from a backup created by the ufsdump command.

=> ufsrestore *options [arguments] [filename ?]*

i Interactive restore.

r Restore the entire backup.

t list the table of contents of the backup

x Restore only the files named on the command line.

a *archive\_file* Take the table of contents information from the named *archive\_file* rather than the tape. Until you actually need to extract a file, the backup volume does not need to be mounted.

b factor Specify the blocking factor for tape reads. By default, the ufsrestore command attempts to figure out the block size of the tape.

f *dump\_file* Use *dump\_file* instead of /dev/rmt/0 as the file to restore from. Typically, the *dump\_file* option specifies a tape or diskette drive.

v Display path names as they are being restored (verbose mode).

> ufsrestore tvf /dev/rmt/0m

> ufsrestore xvf /dev/rmt/0m /etc/passwd

**b15.14 [::restoresymtable]**

The restoresymtable file is created when restoring the entire contents of a dump tape. The restoresymtable file is used for check-pointing, which is information passed between incremental restores. The restoresymtable file is not necessary once the restore is complete and should be removed.

**b15.15 [restore files, How to]**

> cd /var/temp

Display the contents of the tape to verify whether the file is on the tape and to identify the correct path name of the file to be restored.

> ufsrestore tvf /dev/rmt/0

Once you have verified that the file is on the tape, extract the individual file.

> ufsrestore xvf /dev/rmt/0 ./etc/passwd

Specify next volume #:>\_ 1

set owner/mode for '?:' [yn] >\_ n

Type the volume number that contains the file you want. Tape volumes start at 1.

**b15.17 [perform an interactive restore, How to]**

Change to a temporary directory and start the ufsrestore command with the interactive option.

> cd /var/tmp

> ufsrestore ivf /dev/rmt/0

ufsrestore >\_ ls

ufsrestore >\_ cd sbin

ufsrestore >\_ cd /

[Add a file to the list of files to be extracted]

ufsrestore >\_ add .rhosts .wastebasket (Files are marked for restore with an asterisk (\*))

[Toggle the verbose mode to on to display inode numbers.]

ufsrestore >\_ verbose

[Delete a file from the list of files to be extracted]

```
ufsrestore > _ delete .rhosts
[Extract the selected files from the dump volume.]
ufsrestore > _ extract
[Exit the interactive restore once the files are extracted.]
ufsrestore > _ quit
```

**b15.19 [move a file system, How to]**

The following example illustrates how to restore an existing file system onto a new, larger disk partition (or disk).

```
> umount /opt
> fsck /dev/rdisk/c0t3d0s5
> ufsdump 0uf /dev/rmt/0 /opt
> format (To partition a new disk)
> newfs /dev/rdisk/c0t1d0s1
> fsck /dev/rdisk/c0t1d0s1
> mount /dev/rdisk/c0t1d0s1 /opt
> cd /opt
> ufsrestore rvf /dev/rmt/0
> rm restoresymtable
[Add the restored file system to the /etc/vfstab file.]
> fsck /dev/rdisk/c0t1d0s1
[Edit the /etc/vfstab file and reboot or use the mountall command.]
```

**b15.23 [restore the root (/) file system]**

Restoring the root file system is a longer procedure because the special files and command utilities that you need reside in the root file system. Make sure you have a functional backup tape of the root file system.

```
ok > boot cdrom -s (Make sure you have the OS CD loaded)
```

```
> newfs /dev/rdisk/c0t3d0s0, yes
> fsck /dev/rdisk/c0t3d0s0
> mount /dev/rdisk/c0t3d0s0 /a
> cd /a
> ufsrestore rvf /dev/rmt/0
> rm restoresymtable
```

Run the installboot program to create the boot block that resides in the first 15 sectors of the disk. The bootblock program contains a ufs file system reader that loads the ufsboot program into memory.

```
> cd /usr/platform/'uname -i' /lib/fs/ufs
> installboot bootblk /dev/rdisk/c0t3d0s0
> cd /
> umount /a
> fsck /dev/rdisk/c0t3d0s0
> init 6
```

**b16.3 ::mt** - The mt command enables direct tape manipulation.

```
=> mt [-f tape-device-name] command [count]
```

The -f option is used to specify the tape device file name, typically a no-rewind device file name. If the -f option is omitted, the value of the TAPE environment variable is used to determine the tape device to manipulate.

**Commands**

status	Displays status information about the tape device.
rewind	Rewinds the tape.
retention	Rewinds the cartridge tape completely, winds the tape forward to the end of the tape, then rewinds back to the beginning of the tape to smooth out the tape tension.
erase	Erases the entire tape.
fsf	Forward skips count tape files.
eom	Skips to the end of the recorded media.

Note: Only the unique prefix of a command is required. For example, use ret in place of retention.

**b16.4 ::tar**

The tar (tape archive) command enables you to back up single or multiple files in a directory hierarchy.

```
=> tar options [arguments] filename ...
```

**Options**

c	Create a new tarfile using the file names specified on the command line.
t	List the table of contents of the tarfile.
x	Extract the specified files from the tarfile. If no file name arguments are specified, the entire archive is extracted.
f	Use the next argument as the name of the tarfile or the device file /dev/rmt/n. You can also set the

- environment variable TAPE. If tarfile is -, the tar command reads stdin and writes stdout.
  - v Print the file names as they are restored (verbose mode).
  - B Perform multiple reads so exactly enough bytes are read to fill a block (necessary when using tar across the network).
  - p Restore the files with the permissions on tape.
- The tar command is unaware of file systems; however, if you specify a directory as a tar argument, it copies the entire hierarchy below the directory. Use a ./ in front of the directory name so that it can be restored relative to a current working directory.

#### b16.5 **::cpio**

The cpio command creates an archive of single or multiple files by taking a list of names from standard input and writing the archive to standard input and writing the archive to standard output, which is usually redirected to a file or a tape device. It creates directory hierarchies, if necessary. The cpio command is usually used with the ls or find commands to generate archives.

=> [command] | ] cpio *options* [>filename...]

##### Options

One of the o,i, or p options must be specified.

- o Create an archive by reading a list of path names generated from standard input and copy those files to standard output.
- i Extract the archive specified by standard input, which is commonly the result of a previous cpio -o command.
- p Read from standard input to obtain a list of file names. This option is useful for disk-to-disk copying.
- B Set block input/output record to 5120 bytes, but does not apply when using the -p option. The default size is 512 bytes.
- c Read or write header information in ASCII-character format for portability to other platforms.
- H Read or write head information in bar, crc, odc, tar, or ustar format. To be used with the -c option.

[Create an archive of the current directory contents.]

```
> find . -print | cpio -ovcB > /dev/rmt/0
```

[Extract the README file from the cpio archive created above]

```
> cpio -ivcB README < /dev/rmt0 or > cpio -ict < /dev/rmt/0m
```

[Use the find command with cpio to archive called file.list with files that begin with the name file:]

```
> find . -name 'file*' -print | cpio -ovcB > file.list
```

[List the file names from the file.list archive:]

```
> cpio ivt < file.list
```

#### b16.7 **::dd** - Copies entire devices. The dd command converts and copies files with various data formats.

=> dd [option=value]

##### Options

- if=file Specify input file name. Default is standard input.
- of=file Specify output file name. Default is standard output.
- bs=n Set both input and output block size. Default for both is 512 bytes.
- conv=scsii

The dd command is useful when combined with other commands to create or extract tapes from a remote tape device.

**[Create a tape archive on a remote tape drive:]**

```
> tar cvf - scripts | rsh enterprise dd of=/dev/rmt/0
```

**[Extract files from a tape archive on a remote tape drive:]**

```
> rsh enterprise dd if=/dev/rmt/0 | tar xvBpf - scripts
```

#### b17.2 **[:NFS - Distributed File System]**

This module presents the concepts and procedures in configuring the NFS environment using the distributed file systems administration commands.

The Solaris 2.x operating system supports Sun's NFS product. The Solaris 2.x system also provides a distributed file system administration package that contains files and commands used to administer the NFS products.

The NFS file system is a Sun product built on top of TCP/IP, Sun's remote procedure call (RPC), and external data representation (XDR), specifications that have been licensed to many vendors. The NFS environment provides file sharing in a heterogeneous environment, potentially containing many different operating systems, including UNIX, MS-DOS, and VMS systems. NFS has become the industry standard in file sharing.

The Solaris 2.x operating systems supports sharing remote file resources as if they were local files and directories. The sharing of file resources is accomplished through distributed file systems (dfs) - file system types that provide the architectural support for mounting over networks.

##### **[The Benefits of NFS:]**

###### Centralized files

Files are located in a centralized location. One copy of a file is available to many systems simultaneously. This is

especially useful with login directories or common data files. Any changes to a file are immediately available on all systems.

#### Common software

The systems can share a software package. This enables reduced disk requirements for each system, as well as easy updates, since fewer copies of the package need to be updated.

#### Files appear to be local

The file sharing is transparent to the user and to any applications.

### b17.4 **[NFS Terminology]**

[NFS servers]

An NFS file server designates local file resources to be shared with other systems on the network.

[NFS clients]

An NFS client machine mounts file resources that are shared over the network and treats the file systems as if they were local.

### b17.5 **[Sharing and Mounting File Resources]**

NFS Server	NFS Client
------------	------------

<b>Daemons:</b> mountd and nfsd	<b>Daemons:</b> statd and lockd
------------------------------------	------------------------------------

<b>Files:</b> /etc/dfs/dfstab /etc/dfs/sharetab	<b>Files:</b> /etc/vfstab /etc/mnttab
---	---

<b>Commands:</b> share and unshare <u>shareall and unshareall</u>	<b>Commands:</b> mount and unmount <u>mountall and unmountall</u>
---	---

The DFS administration command set contains commands for sharing and mounting file resources.

There are several files used to automatically share and mount remote file resources. The superuser can set up a system to share and mount file resources by adding entries into these files.

### b17.6 **[::NFS Daemons]**

NFS operation requires daemons running on the NFS server and client.

#### **[Mount Daemon]**

When a client issues an NFS mount request, the mount process contacts the server's mount daemon (/usr/lib/nfs/mountd) to get a file handle for the file resource to be mounted. The local mount process then writes the file handle (along with other information about the mounted resource) to the /etc/mnttab file. The file resource is mounted, and the local kernel uses the file handle during all attempts to access the file resource.

File handles are client references that identify a unique file or directory on the server. File handles encode the file's inode number, inode generation number, and disk device number. The client has no way of knowing what may be affecting files and directories on the server.

#### **[The NFS Server Daemon]**

When a process on the client attempts to access the remote file resource the NFS server daemon (usr/lib/nfs/nfsd) running on the server gets the request (along with resource's file handle) and performs the file operation. It then returns any data to the requesting process on the client.

The server daemons are started from the /etc/init.d/nfs.server script. The nfs.server script also defines the number of daemons that are started.

If a system has entries in its /etc/dfs/dfstab file, these daemons are started when the system enters run level 3.

#### **[NFS Daemons]**

Two daemons run on NFS clients. Client daemons are started automatically when a system enters run level 2.

Two daemons, /usr/lib/nfs/statd and /usr/lib/nfs/lockd, run on NFS clients. These daemons work together to provide both crash and recovery functions and locking services in the NFS environment. These daemons typically do not require administrative intervention and are started from the ./etc/init.d/nfs.client script.

### b17.8 **[/etc/dfs/dfstab]**

The /etc/dfs/dfstab file contains share commands. The shareall and unshareall commands are used to explicitly execute the commands in this file.

```
> cat /etc/dfs/dfstab
```

The commands in the /etc/dfs/dfstab file are run when:

- The system enters run level 3.
- The shareall command is run by the superuser (the NFS daemons must be running).
  - The /etc/init.d/nfs.server script (which contains a sharell command) is run by the superuser with the start argument. This script will start the NFS server daemons.

Note: If the nfs.server script does not read any NFS entries in the /etc/dfs/dfstab file, it exits without running the NFS daemons. To see if resources have been successfully shared, use the dfshares command.

### b17.9 **::share**

The `/usr/sbin/share` command makes file resources available for mounting by remote systems. If no argument is specified, then the `share` command displays all file resources currently shared.

`=> [-F nfs] [-o options] [-d description] pathname`

`-F nfs` This option must be used with `share` command entries in the `/etc/dfs/dfstab` file for the `shareall` script to work properly. This option is not required at the command line because `nfs` is the default remote file system type (that is, it is listed as the first file system type in the `/etc/dfs/fstypes` file).

`-o options` Controls clients access to the NFS shared resources.

`-d description` A comment that describes the file resource to be shared. This comment is displayed by the `share` command with no arguments.

`pathname` Specifies the resource to be shared.

`> share -F nfs /usr/man`

The `share` command writes information about all shared file resources into the `/etc/dfs/sharetab` file.

`> share -F nfs -oro /usr/share/man`

[Access Options]

The options to restrict read and write capabilities of NFS clients are:

`ro` Only read requests are accepted by the server.

`ro=client:client` Forms a client list of systems allowed to access the file systems read only.

`rw` Read and write requests are accepted by the server.

`rw=client:client` Forms a client list of systems allowed to access the system read write.

`root=client` Superusers of the specified system of systems are allowed to perform superuser privileged request on the shared file system.

These options can be combined, separated by commas, to form complex restrictions on what systems can access.

By default, NFS mounted directories are available with read and write privileges and no superuser access to all systems.

Notes: The file permissions take precedence over NFS access.

b17.11 **::unshare** - The `/usr/sbin/unshare` command makes file resources unavailable for mounting by remote systems.

`=> unshare [-F nfs] pathname`

Options

`-F nfs` Specify `nfs` as the file system type. This option is not typically required because `nfs` is the default remote file system type.

`pathname` Specify the path name of the file resource to be unshared.

Ex: `> unshare /usr`

b17.12 **::shareall**

The `/usr/sbin/shareall` and `/usr/sbin/unshareall` commands are used to share and unshare multiple resources such as all NFS resources.

`=> shareall [-F nfs]`

Without any arguments, the `shareall` command shares all file resources listed in the `/etc/dfs/dfstab` file. The `-F nfs` option is only required if other file resource types are listed in the `/etc/dfs/dfstab` file along with NFS resources.

**::unshareall**

`=> unshareall [-F nfs]`

Without any arguments, the `unshareall` command unshares currently shared file resources. It does this by reading the `/etc/dfs/sharetab` file. The `-F nfs` option is only required if other remote file resource types (for example, `rfs`) are currently shared along with NFS resources.

Assuming that `/dev/dsk/c0t0d0s0` is a newly created file system not listed in the `/etc/vfstab` file, the following command would work because `ufs` is the default specified in the `/etc/default/fs` file.

`> mount /dev/dsk/c0t0d0s0 /database`

`[::dfshares]`

`=> dfshares [-F nfs] [host]`

Without arguments, the `dfshares` command displays shared resources for the local server.

`> dfshares`

`> dfshares river1`

**::dfmounts** - Displays mounted resource information.

`=> dfmounts [-F nfs]`

Without arguments, the `dfmounts` command displays the shared resource and clients mounting the resource.

`> dfmounts`

b17.14 **[Setup NFS Client-Server environment]**

1. Edit the `/etc/dfs/dfstab` file to enable automatic sharing of resources whenever the system enters run level 3.

`> share -F nfs -o ro=client /usr/share/man`

The `-F nfs` option must be used in `share` command entries in this file for automatic NFS sharing to work

- properly.
2. Start the NFS server daemons by running the a script. This shares the contents of the `/etc/dfs/dfstab` file.
    - > `/etc/init.d/nfs.server start`
  3. Use the `dfshares` command to verify that the resource is available.
    - > `dfshares`
  4. The server keeps a table of clients mounting its resources in the `/etc/rmtab` file. To identify these clients, run the command `dfmounts`.
    - > `dfmounts`

**b17.15 ::mount**

The `/sbin/mount` command is used to attach either a local or remote file resource to the file system hierarchy.

=> `mount [-F nfs] [-o options] server:pathname mount_point`

Ex: > `mount -F pluto:/usr/share/man /usr/share/man (pluto must be in /etc/inet/hosts)`

`-F nfs` Specify `nfs` as the file system type. This option is not required since `nfs` is the default remote file system type.

`-o options` Specify a comma separated list of file system-specific options, such as `rw` to mount the file resource read-write, and `ro` to mount the file resource read-only (the default is `rw`).

`server:pathname` Specify the name of the server and the path name of the remote file resource; these are separated by a colon (:).

`mount_point` Specify the path name of the mount point on the local system (which must already exist).

**[Mount a remote file system]**

> `mount venus:/usr/share/man /usr/share/man`

Note: If the file resource is listed in the `/etc/vfstab` file, you can specify either `server:pathname` or `mount_point` on the command line because the `mount` command checks the `/etc/vfstab` file for more information.

**b17.16 [:/etc/vfstab]**

(To mount remote file systems at boot, place the following in the `/etc/vfstab` file.)

>> `river1:/usr/share/man - /usr/share/man nfs - yes soft,bg`

Fields

`device to mount` The name of the server and the path name of the remote file resource; separated by a colon (:).

`device to fsck` NFS resources are never checked from the client; this field is always null for NFS resources.

`mount point` The default mount point for the file resource.

`FS type` Always `nfs` for NFS resources.

`fsck pass` NFS resources are never checked from the client; this field is always null for NFS resources.

`mount at boot` Either `yes` or `no`, indicating whether the file resource should be mounted when the system enters run level 2 or when the `mountall` command is issued.

`mount options` A comma separated list of mount options.

Mount Options

`rw|ro` The resource is mounted read-write or read-only. The default is read-write.

`bg|fg` If the first mount attempt fails, retry in the background or foreground. The default is retry in the foreground.

`soft|hard` Return an error if the server does not respond, or continue to retry the mount until the server responds. The default is a hard mount.

`intr|nointr` Enable or do not enable keyboard interrupts to kill a process that is hung waiting for a response on a hardmounted file system.

`suid|nosuid` Enable or do not enable `setuid` execution. The default enables `setuid` execution.

`timeo=n 1.1` Set timeout to `n` tenths of a second. The default timeout is 11 1/10 seconds.

`retry=n` Set the number of times to retry the mount operation. The default is 10,000 times.

**b17.19 ::umount**

The `/sbin/umount` command is used to detach either a local or remote file resource from the file system hierarchy.

=> `umount [-F nfs] server:pathname mount-point`

The command line can specify either `server:pathname` or `mount-point`.

Options - [`-F nfs`] Specify `nfs` as the file system type is the default remote file system type.

**b17.20 ::mountall**

=> `mountall -r [-F nfs]`

Without any arguments, the `/sbin/mountall` command mounts all file resources listed in the `/etc/vfstab` file with a `mount-at-boot` value of `yes`. To limit the action of this command to remote file resources, use the `-r` option.

The `-F nfs` option is only required to restrict the action of this command to NFS resources if other remote file resource types (such as `rfs`) are listed in the `/etc/vfstab` file along with the NFS resources.

> `mountall -r`

**::umountall**

=> `umountall -r [-F nfs]`

Without any arguments, the `/sbin/umountall` command unmounts all currently mounted file resources. To limit the action of this command to remote file resources, use the `-r` option

The `-F nfs` option is only required to restrict the action of this command to NFS resources if other remote file resource types are currently mounted along with the NFS resources.

```
> umountall -r
```

#### b17.21 **[setup the NFS client, How to]**

1. Use the `/usr/sbin/dfshares` command to display a server's available resources.

```
> dfshares river1
```

2. Use the `/sbin/mount` command to access the remote file resource.

```
> mount river1:/usr/share/man /usr/share/man
```

The `/usr/share/man` directory on the client is the mount point in the local system's file hierarchy. This directory should be empty.

3. Edit the `/etc/vfstab` file to add an entry for the remote resource that is automatically mounted whenever the system enters run level 3.

```
>> river1:/usr/share/man - /usr/share/man nfs - yes -
```

4. Remote file resources can be unmounted from the client by using the `/sbin /umount` command.

```
> umount /usr/share/man
```

```
>> nfs mount: /usr/share/man: is busy
```

This common error message usually means a user is using the resource if it is an application, or someone is accessing the directory using the `cd` command.

#### b17.22 **[Troubleshooting NFS]**

Most NFS problems are discovered through console messages or symptoms on a client.

##### **[Name to Address Translation Error]**

###### Error Message

```
>> nfs mount: mers:: RPC: Name to address translation failed - n2a: hostname not found
```

This messages displays during the boot process or in response to explicit mount request and indicates an unknown server.

###### Solution

Check that the host name in the hosts database is spelled correctly.

##### **[Server Not Responding Error]**

###### Error Message

```
>> NFS server mars not responding, still trying
```

This message displays during the boot process or I response to an explicit mount request and indicates a known server that is unreachable.

###### Solution

1. Check to see if the server is down.

2. Check to see if the network between your machine and the server is down by using the `ping` command.

##### **[Mount Point Error]**

###### Error Message

```
>> mount: mount-point /DS9 does not exist.
```

This message displays during the boot process or in response to an explicit mount request and indicates a nonexistent mount point.

###### Solution

Check that the mount point exists on the client and is spelled correctly on the command line or in the `/etc/vfstab` file.

##### **[No Such File Error]**

###### Error Message

```
>> nfs mount: mars:/opr: No such file or directory
```

This message displays during the boot process or in response to an explicit mount request and indicates an unknown file resource name on the server.

###### Solution

Check that the directory exists on the server and is spelled correctly on the command line or in the `/etc/vfstab` file.

##### **[RPC Error]**

###### Error Message

```
>> nfs mount: mars: RPC: Program not registered
```

This message displays during the boot process or in response to an explicit mount request and indicates a server that is reachable but is not running one or more of the server daemons.

###### Solution

1. Use the `who -r` command on the server to check whether it is at run level 3. If it is not, change the run level to 3 with the `init 3` command.

2. Use the `ps -e` command on the server to check whether the mount daemon and NFS server daemons are running. If they are not running, start them with the `/etc/init.d/nfs.server` script and the start keyword.

**[Stale File Handle Error]**

Error Message

>> stale NFS file handle

This message displays when a process attempts to access a remote file resource and the file handle is out of date.

Solution

The file resource may have been moved on the server. Unmount and mount the resource again on the client. You may get the message >> `nfs mount: mars:/usr/share/man: No such file or directory`. In this case, contact the administrator of the server and ask about the lost file resource.

**[No Cable Error]**

Error Message

>> `le0: No carrier - transceiver cable problem?`

This message displays during the boot process or in response to an explicit mount request and indicates a network problem.

Solution

Check the physical network connections between your machine and the server, including terminators.

b17.26 **[NFS Commands, Files, and Daemons]**

	<b>NFS Server</b>	<b>NFS Client</b>
Commands	<code>share directory</code> <code>unshare directory</code> <code>shareall</code> <code>unshareall</code> <code>dfmounts</code> <code>/etc/init.d/nfs.server</code>	<code>mount server:directory mount-point</code> <code>mountall -r</code>  <code>umountall -r</code> <code>dfshares server</code> <code>/etc/init.d/nfs.client</code>
Files	<code>/etc/dfs/fstypes</code> <code>/etc/dfs/dfstab</code> <code>/etc/dfs/sharetab</code> <code>/etc/rmtab</code>	<code>/etc/dfs/fstypes</code> <code>/etc/vfstab</code> <code>/etc/mnttab</code>
Daemons	<code>/usr/lib/nfs/nfsd</code> <code>/usr/lib/nfs/mountd</code>	<code>/usr/lib/nfs/statd</code> <code>/usr/lib/nfs/lockd</code>

b18.3 **[::Automount]**

*File Systems are mounted on demand*

File systems can be automatically mounted to share resources more efficiently.

*File Systems are unmounted automatically.*

The remote file resource remains mounted for as long as it is being used. If none of the files or directories in the hierarchy are accessed within a specific time-out period (the default is 5 minutes, the automount automatically unmounts the resource.

*Automounting does not impact the server side of NFS*

It does not matter to the NFS server whether the shared directories are being accessed through the mount command or the automount.

**[How automount works]**

The client autofs file system

File systems shared through NFS software can be mounted automatically.

The autofs file systems are defined in automount maps located in the `/etc` directory on the client system. Once the autofs mounts are set up, they can trigger file systems to be mounted under them. When automount, the program that monitors automounting, receives a request to access a file system that is not currently mounted, it calls `automountd`, which actually mounts the requested file system.

The automount program

The automount program, called at system startup time, reads the master map file `auto_master` to create the initial set of autofs mounts. These autofs mounts are not automatically mounted at startup time. They are points under which file systems are mounted upon demand.

The automountd daemon

The `automountd` daemon is started by the automount program and mounts file systems on demand.

The `automountd` daemon is completely independent from the automount command. Because of this separation, it is possible to add, delete, or change map information without having to stop and start the `automountd` daemon process.

b18.5 **[::automount]**

When making changes to the master map or creating a direct map, make the change effective by running the

automount command.

=> automount [ -t duration] [ -v]

#### Options

-t duration This option enables you to specify a time, in seconds, that the file system remains mounted when not in use. The default is 5 minutes.

-v Verbose mode, displays output as the command runs.

There is no need to stop and restart the automount daemon because it is stateless. Existing entries in both direct and indirect maps can be modified at any time. The new information is used when the automountd command next uses the map entry to perform a mount.

Master map not stateless. Type automount to re-read the map.

### b18.6 **[::autofs Maps]**

The autofs files, called maps, identify the file system resources to be automatically mounted. There are three map types:

**1.Master Map** - The master mapping read by the automount command. The function of this map is to list the other maps used for establishing the autofs file system.

In > /etc/auto\_master

**2.Direct Map** - A map that lists the mount points as full path names. The function of this map is to explicitly indicate the mount point on the client. auto\_direct must be built.

In > /etc/auto\_direct

**3.Indirect Map** - Map that lists the mount points as relative paths. The function of this map is to use a relative path to establish the mount point on the client.

In > /etc/auto\_home

Automount maps are implemented as ASCII data files, NIS, or NIS+ database files. Together, these maps describe information similar to the information specified in the /etc/vfstab file for remote file resources.

Note - The plus (+) symbol directs the automounter to look at NIS+ databases. Comment the line out to use only local files.

### b18.8 **[The Master Map]**

#### Fields

*mount\_point* This is the full path name of a directory. If the directory does not exist, autofs creates one if possible

*map\_name* The name of a direct or indirect map. These maps are directions to mounting information.

*mount-options* The general options for the map. The mount options are the same as those for standard NFS mounts.

#### **Direct Map Entries**

The / - entry defines a mount point for a direct map. The mount point / - is a pointer that informs the automounter that the full path names are defined in the /etc/auto\_direct file.

#### **Indirect Map Entries**

The /net, /home, /xfr entries define mount points for indirect maps. The maps -hosts, auto\_home, and -xfr list relative path names only. Indirect maps get the initialize path of the mount point from the master map.

#### **The -hosts map**

This entry specifies that all shared resources from each NFS server listed in the hosts database (/etc/inet/hosts, NIS or NIS+) are made available under the directory /net/host. This is a default entry and mounts every file system shared by the NFS server under the directory /net/host.

The -hosts map is a generic map that is useful as a general purpose access to any NFS server. If an NFS server has 20 directories exported, a reference to that NFS server's directory within /net would mount all 20 directories.

Note: The -xfr map is the primary service provided by a federated naming service and is not discussed.

### b18.10 **[Direct Maps]**

Direct maps specify the full path name of the mount point, the specific options for this mount, and the name of the server sharing resource.

#### Fields

*key* The full path name of the mount point.

*options* The specific options for a given entry.

*location* The location of the file resource specified in *server:pathname* notation.

The following direct map entry specifies that the client mounts the /usr/share/man directory as read-only from the servers sun, moon, or stars.

```
>> /usr/share/man -ro sun,moon,stars:/usr/share/man
```

This entry uses a special notation, a comma separated list of servers, to specify a powerful automounter feature-multiple locations for a file resource.

### b18.12 **[Indirect Maps]**

The indirect maps have relative paths in the key field. The first part of the path name is specified in the master

map. Indirect maps are useful when you want to mount many remote file resources below a common directory.

#### Fields

**key** The full path name of the mount point relative to the beginning of the path name specified in the /etc/auto\_master map.

**options** The specific options for a given entry.

**location** The location of the file resource specified in *server:pathname* notation.

#### *The auto\_home Indirect Map*

The auto\_home indirect map provides a consistent view of home directories across the network, regardless of which system a user is currently logged into.

#### The Substitution String for an Indirect Map

The following entry reduces the auto\_home file to a single line. The use of substitution characters specifies that for every login ID, the client remotely mounts the /export/home/loginID directory from the NFS server mars onto the local mount point /home/loginID.

```
> * mars:/export/home/&
```

This entry uses the wildcard character (\*) to match any key, and the substitution character (&) to substitute for the current key, at the end of the location specification.

#### b18.14 **[setup a direct map, How to]**

This procedure describes how to setup an application - the Solaris 2.x man pages - using the automount daemon.

1. Edit the /etc/auto\_master file by placing comment symbols (#) in front of the +auto\_master entry.

2. Add a direct map entry. This entry instructs the automounter to look in the /etc/auto\_direct file for direct map entries.

```
> / -auto_direct
```

3. Create a new file called /etc/auto\_direct and add the following entry for the directory you want to automount.

Replace server with the host name of the server.

```
> _ /usr/share/man -ro server:/usr/share/man
```

4. Make the changes effective.

```
> automount -v
```

#### b18.15 **[setup an indirect map, How to]**

1. Create an /etc/auto\_patch map. Enter the patch directory name on the client and the *server:resource*.

2. Edit the /etc/auto\_master file. Place comment symbols (#) in front of the +auto\_master entry. Add the patch directory and map.

3. Create the /etc/auto\_patch map.

```
> patch mars:/export/patch
```

4. Make the changes effective.

```
> automount -v
```

#### b20.2 **[::NIS+ Environment]**

##### **[The NIS+ Service]**

NIS+ is a service that provides information about users, workstations, and network resources. It makes this information available to users and applications that request it. It also provides security measures to protect against unauthorized access.

NIS+ is not just an enhanced version of NIS. It is a new application that performs the same function as NIS but with many different features. For example, it uses secure rpc, and it can be updated from any authenticated and authorized user of system.

Completely backwards-compatible with NIS.

*NIS stores ASCII text files*

```
Host IP
```

```
IP Host
```

NIS+ uses Relational Database Technology

Solaris

NIS+ (Root Master)

IBM - NIS      HP - NIS      DEC - NIS

(NIS send all files to replica servers)

#### b20.4 **[The NIS+ Namespace]**

The NIS+ *namespace* is a hierarchical structure in which the NIS+ information is stored. Each namespace has a root master server that serves the root domain at the top of the namespace.

It is similar in structure to the Solaris file system, but you cannot access it with Solaris file commands. The NIS+ namespace is accessed with NIS+ commands.

The NIS+ namespace consists of domains. Each domain is a unique entity within the namespace, again very similar to a directory in the Solaris file system.

The org\_dir directory stores the NIS+ table objects.

The groups\_dir directory stores NIS+ group objects.

b20.6 **[NIS+ Objects]**

**[NIS+ Directory Objects]**

The main components of the namespace. They contain other directory objects, table objects, and group objects.

**[NIS+ Table Objects]**

NIS+ table objects store the information in the NIS+ namespace. The Solaris 2.x environment provides 16 types of tables, each of which stores a different type of information about users, workstations, or resources on the network. A set of NIS+ tables stores information for that particular domain only.

**[NIS+ Group Objects]**

NIS+ group objects are used for NIS+ security. An NIS+ group is a collection of users and workstations that are identified by a single name and are used to facilitate NIS+ security. Although NIS+ groups are optional, they are a security convenience, enabling the administrator to assign the same access rights to a group of users and workstations.

**[Table Objects of a Domain]**

*The NIS+ domain can contain the following table objects:*

---

auto_home	ethers	networks	rpc	timezone
auto_master	group	netmasks	services	
bootparams	hosts	protocols	sendmailvars	
cred	netgroup	passwd	mail_aliases	

b20.8 **[Naming Conventions]**

The NIS+ object names are formed by appending the root directory name (including the period) to their names.

This is called a fully qualified name. For example, eng.solar.com. and fin.solar.com. are both fully qualified names that represent the eng.solar.com. and fin.solar.com. subdomains within the solar.com. domain.

Like UNIX files and directories, NIS+ objects can be referred to by their full or partial names.

**[Root Domain Names]**

Root domain names must contain two components followed by a dot, as follows: solar.com.

**[Fully Qualified Names]**

A fully qualified name is the complete name of the component. It is formed by starting with the name of the component and appending the names of all involved components up to the root domain, separated by dots (.). For example, the fully qualified name of the sales division's hosts table at Solar, Inc., is as follows:

Hosts.org\_dir.sam.solar.com.

**[Partially Qualified Names]**

A partially qualified NIS+ component name is analogous to a relative path name and is simply the name of the component. For example, the hosts table's partially qualified name is as follows: hosts

b20.10 **[The NIS+ Model]**

art/sa285\_20-10.gif

NIS+ solves the problem of having common information available to multiple machines by storing this information in databases on server machines. The servers make the information available to clients on demand.

**[NIS+ Servers]**

A server is a process that receives client process requests, looks up requested information in a database, and returns the information to the client process. The term server is also applied to machines that run server processes and store databases on their disks. Every domain is served by one master server and one or more replica servers.

**[Master Server]**

A master server contains the master set of database information in the form of tables. Changes are made to these tables and are automatically pushed to the replica servers.

**[Replica Server]**

A replica server maintains copies of the tables to distribute the burden of answering client requests and provide backup sources of information in case the master server is down.

Both master and replica servers store NIS+ table information and answer client requests. Only the master, however, stores the master copy of the tables. The replicas store only duplicates of the master.

**[NIS+ Clients]**

A client is a process or machine that sends requests for information to the network. These processes use calls to the RPC library to make requests.

b20.12 **[NIS+ Master Servers]**

**[Root Master]**

At the top of the NIS+ namespace or root domain is the root master, which is a client of the root domain because there is no domain above it. There can only be one root domain.

**[Domain Master]**

A domain master server in a domain contains the master set of database files for that domain. These database

files contain different and unique information for that domain.

### **[NIS+ Servers as Clients]**

An NIS+ server is also a client of its parent domain. The domain the server belongs to is always above the domain it serves, except the root domain server. The server supporting the root domain belongs to the root domain.

### **[Replica Servers]**

NIS+ domains are supported by one master server and one or more replica servers.

- Reliability - One advantage of having replica servers is reliability. If one server is unable to handle a request, one of the other servers can reply.
- Efficiency - An additional advantage is efficiency. If there are many requests, they can be better handled by multiple servers.
- Simplicity - Another advantage is simplicity for the system administrator. The administrator creates the table information in one location (the master server), and the master server propagates it to the replica servers. Similarly, updates are made to the master server, and the master server propagates the updates to the replica servers.

### **[NIS+ Clients]**

Objects in the NIS+ namespace are stored on NIS+ servers. The servers provide information to the clients that request it. Every NIS+ domain specifies a list of servers that stores the information in its domain.

An NIS+ client belongs to an NIS+ domain. When a client is first initialized, its domain is identified and the name is then written to the `/etc/defaultdomain`. At that time, a `coldstart` file is created for the client. This file lists all the NIS+ servers that support the client's domain. When a client sends a request in its domain, it sends it to one of these supporting servers.

### b20.14 **[NIS+ Security]**

#### **[NIS+ Principals]**

NIS+ security protects the information from unauthorized access. Access is granted only for NIS+ principals.

An NIS+ principal is one of the following:

- A user logged on to an NIS+ client.
- A user logged in as root on an NIS+ client.

For NIS+ security purposes, a user logged in as root is considered to be the workstation itself. Thus, an NIS+ principal is either a user or a workstation.

#### **[NIS+ Security Privileges]**

NIS+ security privileges are assigned to NIS+ principals in two stages:

- Authentication - Credentials identify a principal. Credentials are stored in a domain's cred table.
- Authorization - Each object in the namespace assigns access rights to different categories of NIS+ principals.

This security information is stored in the object definitions.

When a principal requests access to an object, the NIS+ server finds out what access rights are assigned to that principal by that particular object and responds accordingly.

#### **[NIS+ Security Overview]**

When an NIS+ server receives a request from an NIS+ client, it first identifies the principal. Then it finds the object the principal wants to access and determines whether the principal has proper access to that object. If the object's definition indicates that the principal has the proper access rights, the server grants it.

### b20.16 **[NIS+ Authentication]**

Authentication is the process of identifying the principal making the request to the NIS+ server. The purpose of authentication is to obtain the principal's name so that its access rights to an object can be looked up (the authorization process).

An NIS+ server authenticates a principal by checking its credentials. NIS+ accepts two types of credentials:

- Local credentials - Credentials are used to map a client's UID to its NIS+ principal name. They are created by extracting the client user's UID and GID from the password record and group tables and placing them in the domain's cred table.
- DES credentials - Credentials generated by creating an additional password (or key) that is required to authenticate the principal. If this additional key is not provided, the principal is considered unauthenticated and is denied access to the object. Usually, the principal's login password and DES key are the same.

#### **[The Credential Table]**

The information for authenticating NIS+ principals is stored in the cred table. There is one cred table for each NIS+ domain. It stores authentication information for the NIS+ principals that want to access that particular domain.

#### **[NIS+ Authorization]**

For the purpose of authorization and the granting of access rights, NIS+ classifies principals into four categories.

<b>Category</b>	<b>Description</b>
Owner	A single NIS+ principal
Group	A collection of NIS+ principals

World	All principals authenticated by NIS+
Nobody	Unauthenticated principals

Access rights are displayed as a list of 16 characters. These access rights are specified as part of an object's definition.

Nobody | Owner | Group | World  
r - - - r m c d r m - - r - - -

An NIS+ group is one or more NIS+ principals grouped as a security convenience. Information about NIS+ groups is stored in NIS+ group objects, under the groups\_dir subdirectory of every NIS+ domain. Note that it is not stored in the NIS+ group table - that table stores information about UNIX groups.

Access rights can be displayed using the nisl or niscat commands, which are covered in the next module.

NIS+ authorization is the process of granting NIS+ principals access rights to an NIS+ object. There are four types of access rights.

**Access Right Description**

Read	Principal can read the contents of the object.
Modify	Principal can modify the contents of the object.
Create	Principal can create new objects in a table or directory.
Destroy	Principal can destroy objects in a table or directory.

NIS+ access rights are similar to regular file permissions.

b20.19 **[NIS+ Security Levels]**

The implementation of the authentication scheme described previously is determined by the domain's level of security.

An NIS+ server can operate at one of three security levels: 0,1, or 2.

These security levels determine the degree to which the principal's credentials are checked.

**Security Level Description**

- 0No checking of the principal's credentials is done. Any client can perform any operation. This level is designed for testing and setting up the initial NIS+ namespace.
- 1Checks the principal's credentials and accepts either LOCAL or DES authentication. Because LOCAL credentials are easily forged, do not use it on networks to which untrusted servers may have access.
- 2Checks the principal's credentials and accepts only DES authentication. This is the highest level of security currently provided and is the default level assigned to an NIS+ server.

The NIS+ service daemon called rpc.nisd that runs on an NIS+ server is started from the /etc/init.d/rpc script. The default security level is 2.

**[Name Service Switch Configuration Files]**

Four versions of the name service switch configuration file are included with the Solaris 2.x release:

1.[:/etc/nsswitch.conf]

This is the default configuration file that specifies the name service that was selected during system installation. It is the source file searched for network information.

2.[:/etc/nsswitch.files]

This is an alternate name service switch file that only searches the local system's /etc files.

3.[:/etc/nsswitch.nis]

This uses the NIS database as the primary source of all information except the passwd, group, automount, and aliases maps, which use the local /etc files first and then the NIS databases. Because the search order for the passwd and group files is the local files first and then the NIS database, there is no need for a plus (+) in the passwd file.

4.[:/etc/nsswitch.nisplus]

This uses NIS+ as the primary source for all information except the passwd, group, automount, and aliases tables, which use the local /etc files first and then the NIS+ databases.

The default /etc/nsswitch.conf file is determined by what name service was selected during installation. The other switch files (listed above) can be copied to the /etc/nsswitch.conf file when changing your name service configuration.

**[:/etc/nsswitch.conf]**

The name service switch function enables NIS+ clients to obtain information from one or more sources such as the local /etc files or the NIS+ tables.

The /etc/nsswitch.conf file, or the name service switch configuration file, contains a list of 15 types of databses, their sources, and the order in which theses sources are searched.

One or more sources can be specified for each database.

**Source Description**

files	The client's local /etc files
nisplus	An NIS+ table
nis	An NIS map

compat Supports old-style "+" syntax for passwd and group information  
dns Applies only to the hosts entry

## b20.23 [Name Service Switch Status and Action Values]

### Return Status

Each source returns a status code that returns a value to the user requesting NIS+ information.

<b>Status Code</b>	<b>Description</b>
SUCCESS	Found the requested entry.
UNAVAIL	Source was unavailable (down).
NOTFOUND	Source contains no such entry.
TRY AGAIN	Source returned "I'm busy, try later" message.

### Actions

For each status code, two actions are possible.

<b>Action</b>	<b>Description</b>
continue	Try the next source.
return	Stop looking for the entry.

The default actions are:

SUCCESS = return

UNAVAIL = continue

NOTFOUND = continue

TRY AGAIN = continue

Ex: >> hosts: nisplus [NOTFOUND=return] files

Assuming that the name service is running, this syntax means that only the NIS+ hosts table is searched.

Remove the [NOTFOUND=return] entry if you want to search the NIS+ hosts table and the local hosts file.

## b21.3 [NIS+ Client Commands]

### [NIS\_PATH Variable]

The NIS\_PATH variable can be set to simplify references to the table names. You can specify tables without the .org\_dir suffix. Type the following commands to set the NIS\_path variable in the Bourne or Korn shell.

```
> NIS_PATH='org_dir.$:.'
```

```
> export NIS_PATH
```

Note: The \$ is an NIS+ metacharacter that expands to the domain name.

**::nisl** - Lists the objects of an NIS+ directory.

```
=> nisl [-l] [directory_name]
```

```
> nisl -l
```

The columns in the listing are: Type Permissions Owner's principal name Date of creation Object name

**::niscat** - Displays the contents of NIS+ tables.

```
=> niscat [-h] tablename
```

### **::nisgrep**

The nismatch and nisgrep commands enable shell scripts to search NIS+ tables. The nisgrep command differs from nismatch in accepting regular expressions for the search criteria rather than simple text settings.

```
=> nisgrep colname=keypat tablename
```

```
Ex: > nisgrep 'uid=11 [234]' passwd
```

### **::nismatch**

```
=> nismatch key tablename
```

```
> nismatch rimmer passwd
```

nismatch is much more efficient than the previous two commands as it does not require the whole table to be copied across the net.

## B21.7 **::passwd**

The passwd command changes entries in the NIS+ passwd table. It modifies local information stored in /etc/shadow files or the NIS+ passwd table depending on settings in the /etc/nsswitch.conf file. It uses secure RPC to communicate with the NIS+ server, and it never sends unencrypted passwords over the network.

```
=> passwd
```

Note: Solaris 2.4 and earlier required => nispasswd.

### **::nisdefaults**

A convenient way for system administrators to display default NIS+ values such as current principal name, domain name, host name in the domain, and so on.

```
?? > nisshowcache -v | more
```

## b21.9 [NIS+ Server Scripts]

Three scripts under the directory /usr/lib/nis are used in the initial setup of NIS+: nisserver, nispopulate, and nisclient. These scripts set up a secure NIS+ domain using level-2 security (DES).

The nisserver script can be used to set up the root master and replica NIS+ servers.

### **[Create a root master]**

Run on the system that is to be the root master, the following will create the root master for the domain sun.edu.

```
> nisserver -r -d solar.com
```

### **[Create a replica server]**

Run from the root master, the following will make the system earth into a replica server.

```
> nisserver -R -d solar.com. -h earth.solar.com.
```

### **[Create a non-root master]**

Run from the root master, the following will create non-root master for the domain soft.solar.com.

```
> nisserver -M -d soft.solar.com. -h pluto.solar.com.
```

### **[Populate NIS+ tables]**

Run on the root master, the following will populate the tables with the contents of the system files in the /etc directory.

```
> nispopulate -F -p /etc
```

(Make sure /etc/hosts are updated first)

### **[Create an NIS+ client]**

Run on the client system, the following makes that system a client of the sun.edu. domain where the root master system name is palau at the IP address of 192.9.200.25.

```
> nisclient -l -h earth -a 192.9.200.25 -d solar.com.
```

## **b21.11 [create a root master, How to]**

1.Login as root on the designated NIS root master, and set your path to include the NIS+ scripts directory.

```
> root_master# PATH=/usr/lib/nis:$PATH;export PATH
```

2.Run the nisserver script to create a root master server.

```
> nisserver -r -d domain.name (When asked to verify type (y) Yes).
```

The script automatically runs the nisinit and nissetup commands and starts a related process.

(This script does the following: > rm -r /var/nis; rm /etc/defaultdomain; reboots.)

3.You are prompted for a network password required by NIS+ for security purposes. Type the root password.

The script updates NIS+ security information and restarts the related process.

4.Use the nispopulate script to populate the newly created NIS+ tables using administration files in the /etc directory. (You may want to edit these files first.)

```
> nispopulate -F -p /etc
```

Note: Be sure to include the host names and IP addresses of the designated NIS+ replica server and client in the /etc/inet/hosts file prior to running the script.

5.You are asked to verify the setup, type (y).

6.The source administration files are listed, and you are prompted once more to verify the creation of the destination NIS+ tables, type (y).

7.The root master is ready.

## **b21.14 [create an NIS+ client, How to]**

Login to the designated client as root, and use the nisclient script to initialize the client.

```
> /usr/lib/nis/nisclient -l -h root_master_hostname -a root_master_IP_address -d domain.name
```

```
> (y), nisplus, password
```

[create an NIS+ replica server, How to]

1.Setup a machine as a client

2.On the client designated to be a replica server, login as root and start the NIS+ service daemon, /usr/sbin/rpc.nisd to setup the replica server.

```
> rpc.nisd
```

3.On the root master, run the nisserver script to initialize the replica.

```
> nisserver -R -d domain.name -h replica_server (Verify when prompted by typing (y).)
```

### **[Automounting a User's Home Directory]**

The following steps describe the procedure for automounting a user's home directory using AdminSuite to update the NIS+ auto\_home table.

- Use the Solstice AdminSuite to create a new user and select the AutoHome setup check box.
- Use the Database Manager to add an entry for an existing user to the auto\_home table.
- Make sure that the user's home directory is shared on the system that contains the user's home directory.
- Have the user log in to a system other than the server than contains the home directory to see if the home directory is mounted.

### **[setup a NIS+ user, How to]**

1.Start the Solstice AdminSuite and click on the User Account Manager icon.

2.Set the Naming Service to NIS+.

3.Click on Apply.

4.Choose Add from the Edit menu.

5.Fill out the Add User form to create a new user and select the AutoHome Setup check box.

6.Click on Add.

7.Dismiss the User Account Manager window.

Now try logging in as the new user on another NIS+ client machine to see if the home directory is automatically mounted.

### b22.3 [::ASET]

The Solaris 2.x environment provides the Automated Security Enhancement Tool (ASET) as an aid to evaluating and enhancing system security features.

The ASET tool is an easy-to-use security product providing automated security administration. ASET can be configured for three security levels: low, medium, and high.

The ASET tool is a simple but powerful tool for users who want security assurances but do not have the time to check for individual security breaches on a daily basis.

Make sure the SUNWast software package is installed before trying to use the ASET software by issuing the pkginfo command.

```
> pkginfo |grep SUNWast
```

ASET provides administrators with options to easily specify three overall security levels.

- Low-level security** - This level provides number of checks, and reports are generated outlining any potential security weakness. Ownership and permissions on important system files are changed to match their setting when a system is first installed.

- Medium-level security** - This level can modify some system files to restrict system access if security risks are found. The modifications should not affect any system services.

- High-level security** - This level provides a secure system by setting system parameters to minimal access permissions. Most system applications and commands should work normally, but security protections take precedence above any other system behavior.

[ASET Tasks]

ASET performs seven tasks, each making specific checks and adjustments to system files and permissions to assure system security. Every task prints a report noting weaknesses found and changes made.

<b>Task</b>	<b>Report Name</b>
Verify that a router can be used as a firewall.	firewall.rpt
Check initialization files (.profile, .login, .cshrc) for umask and PATH variable settings.	env.rpt
Check the contents of system configuration files such as /etc/default/login.	sysconf.rpt
Check the consistency and integrity of /etc/passwd and /etc/group entries.	usrgrp.rpt
Verify appropriate system file permissions.	tune.rpt
Examine owner, permissions, links, and size of important system files.	cklist.rpt
Verify appropriate EEPROM security parameter.	eeeprom.rpt

At least seven tasks are run at each security level. See the tune.low, tune.med, and tune.high scripts in the /usr/aset/masters directory to identify potential system changes by each task at the different security levels.

ASET generates reports based on the tasks that have been performed. These reports are located in the /usr/aset/reports/latest directory.

### b22.8 [Firewall]

A router is any machine that has two Ethernet (network) interfaces whose purpose is to pass information from one network to another. A firewall machine is a router that does not forward data or advertise routes from one network to the other. This means users must login to the firewall system to gain access to the other network.

The firewall task script in the /usr/aset/tasks directory is run at every level, but only takes action at the highest security level. The two actions taken by the firewall task script are:

- Turns off IP forwarding to ensure that the firewall system does not send information from one network to another.
- Ensures the in.routed daemon is started with the -q flag to prevent broadcasting visible routing information.

#### [Eliminating the Firewall task]

If your system does not require firewall protection and you want to run at the highest security level, you can eliminate this task by editing the asetenv file in the /usr/aset directory to remove the firewall task from the TASKS list.

**::aset**

Use the aset command to check your system's security. This command provides a task report when completed.

The security check for all seven tasks is run at the low level by default.

The system runs considerably slower for approximately 15minutes while the aset processes are running in the background.

```
> /usr/aset/aset
```

To set security to the highest level, use > aset -l high.

**::taskstat**

Use the /usr/aset/util/taskstat command to find out whether the aset command has finished performing each of

the seven task checks.

> /usr/aset/util/taskstat

b22.11 **[::usr/aset/reports/latest/taskstatus]**

You can also view this file to verify that all tasks are done.

**[Restoring pre-ASET system files]**

When ASET is run for the first time, it saves and archives the original system files in the /usr/aset/archives directory. To restore these system files, use the aset.restore command.

=> aset.restore [ -d aset\_dir]

Options

-d aset\_dir Specify the working directory for ASET. By default, this directory is /usr/aset.

Ex: > /usr/aset/aset.restore

b23.3 **[Serial Devices]**

A serial device such as a serial printer, modem, or terminal is communications hardware that transfers data in a serial fashion - one bit after another.

Another type of communications hardware is a parallel device that transfers one or more bytes simultaneously, such as a Centronics printer.

Telecommunications lines are usually serial, which is why modems are connected to a computer by a serial port.

**[Serial Ports]**

A port is a pathway on a computer that is used to connect communication lines and modems. A port is made up of both hardware (pins and connectors) and software (device driver).

Types of ports include serial, parallel, small computer system interface (SCSI), and Ethernet.

A serial port uses a standard communications protocol to transmit a byte of information bit-by-bit over a single line. Sun serial ports use RS-423/RS-232-C communications protocols standards for serial interfaces between computers and peripheral devices.

Sun workstations usually come with two serial ports that can be used to provide access to different serial devices such as: Terminals, Modems, and Printers.

(As you look at the back of a workstation the serial ports are labeled: TTYA TTYB )

**[Serial Printers]**

Connecting serial printers is the same as connecting terminals and modems. All three need an associated service to operate.

The main way to gain access to a service is through a port monitor program that continuously "listens" for requests such as login or remote print requests.

**[Standard RS-232-C Interface]**

**Workstation or Terminal**

art/sa285\_23-08.gif

**Data Terminal Equipment**

**Modem**

**Data Communication Equipment**

Data terminal equipment (DTE) refers to terminals, workstations, and usually printers. DCE usually refers to modems, multiplexors, and data switches. The primary determinant of whether a device is considered a DTE or a DCE is which pins on the RS-232-C interface expect which signals. A DTE device transmits on pin 2 and receives on pin 3. A DCE device transmits on pin 3 and receives on pin 2.

Sun's current workstation product line uses the RS-423 interface, which maintains compatibility with the RS-232-C 25-pin connector.

b23.9 **[Null Modem Cable]**

art/sa285\_23-08.gif

A null modem cable provides the means for two DTE devices to communicate without the need for a modem. The direct-wire approach crosses pins 2 and 3 so that transmitted and received data are correctly connected between the two DTE devices. The ports on the CPU board can be connected to terminals and serial printers with the null modem cable. Only pins 2,3, and 7 are required for communication between the devices.

b24.2 **[::SAF - Service Access Facility]**

SAF is a suite of commands used to provide access to serial devices such as terminals and modems. In addition, SAF provides access to network services, such as remote print requests.

The Solaris 2.x environment uses STREAMS-based character I/O for terminal and network devices. A STREAMS module called ldterm is used to perform standard terminal I/O processing. The ldterm module is also known as the line discipline module.

There are four primary commands for SAF. This module will focus on the first three.

**Command Name Description**

sacadm Sac administrative command used to add, remove, disable, and monitor port monitors.

pmadm Service administration, used to associate a port monitor instance with the service it provides.

ttyadm	Provides ttymon-specific information, such as the port device name, to the pmadm command.
Nlsadmin	Provides listen-specific information, such as the server providing the service, to the pmadm command.

---

### [SAF Overview]

SAF provides two basic services:

- Management of serial ports.
- Monitoring remote printer requests on network ports.

SAF consists of the service access controller (sac), a controlling daemon, know as a port monitor, and the supporting directory structure.

sac oversees management of all SAF processes. It is started at bootup.

A port monitor is a process on a system that is responsible for monitoring incoming serial and network ports for requests and dispatching them appropriately. There are two daemons directly responsible for management of ports:

- ttymon** manages serial ports
- listen** manages network ports

System administrators configure SAF to provide services to serial ports or to provide “listening” for remote printer requests. This configuration requires setting up the appropriate daemon to monitor the appropriate serial or network port.

SAF has been designed to handle large numbers of serial ports and printer requests. To handle this volume, a hierarchical structure of files and administration commands are provided to control devices individually, and in groups.

#### b24.4 [The SAF File Structure]

art/sa285\_24-04.gif

The basis of SAF are service tags associated with port monitors. There are two types of service tag, stored in a hierarchical structure of directories, and two port monitors.

#### [Serial Port Service Tag]

The serial port service tag defines the connection between the I/O sbus of the system, and the serial port. It also defines the services to be provided. This definition includes: Baud Rate, Terminal type, Streams module, Login service (/usr/bin/login)

There are two services tags setup for the existing serial ports: TTYA and TTYB

Other service tags are identified with names assigned by the system administrator.

Service tags are stored in files called \_pmtab (port monitor table).

#### [Printer Service Tag]

The printer service tag defines which OS (Sys5 or BSD) printer daemons to use for dispatching requests. Printer service tags are also kept in files called \_pmtab.

#### b24.6 [Port Monitor Tables]

Services tags are stored in files called \_pmtabs (port monitor tables). These tables facilitate management of the serial or network ports by the port monitors. They also enable the system administrator to group homogenous devices together.

#### [Port Monitor Tags]

Port monitor tables are, in turn, kept in directories known as pmtags (port monitor tags).

**zsmom** is an established directory (pmtag) containing predefined service tags (in \_pmtab) for the two standard serial ports /dev/term/a and /dev/term/b. The service tags are called ttya and ttyb.

**tcp** is the default directory (pmtag) for remote printer requests. This pmtag is setup when printers are configured.

Note: The sys admin may establish many different \_pmtab files in other pmtags (directories).

#### [Service Access Controller Table]

Port monitor tables (\_pmtab) must have the appropriate port monitor (daemon) associated with them. Serial ports require the ttymon daemon, and network ports require the listen daemon.

Definitions are placed in the \_sactab (service access controller table) located in the /etc/saf directory to initialize the appropriate port monitor with the corresponding port monitor table. The initialization process uses the port monitor tag (pmtag) to identify the port monitor with the port monitor table.

#### [Port Monitors]

The port monitors are daemons that are responsible for monitoring incoming ports (either serial or network), to detect incoming service requests (such as login requests or print requests), and to dispatch them appropriately.

#### [::ttymon Monitor]

The ttymon daemon is a STREAMS-based TTY port monitor. It monitors serial ports, sets terminal modes and baud rates, and invokes the login process or any other specified service.

#### [::listen Monitor]

The listen daemon is the network listener port monitor. It listens on network ports for service requests such a remote print requests. The listen port monitor then processes the requests by invoking other print daemons to

provide the services.

#### b24.8 **[The Initialization Process]**

art/sa285\_23-08.gif

The init process spawns sac, the master process of SAF. sac is started when a system enters run level 2 by an entry in the /etc/inittab file. There is a separate entry to start a ttymon port monitor for the system console.

#### **[The Bootstrap Sequence]**

- 1.The init process reads the /etc/inittab file and starts up sac at run level 2.
- 2.When sac is invoked, it reads a per-system configuration script (/etc/saf/\_sysconfig). This file is empty by default but can be used to customize the environment, for example, setting time zones for terminal in different geographical locations.
- 3.The sac reads its administrative file (/etc/saf/\_sactab) and starts up the required port monitors as specified.
- 4.Each port monitor reads its own /etc/saf/pmtag/\_pmtab file to identify the services to be started.
- 5.The service tag entry tells the port monitor what service to start for each port.

#### b24.10 **[Terminal Initialization Process]**

Once a ttymon monitor instance is invoked by sac, it starts to monitor its port, using the following process:

- 1.The ttymon monitor first initializes the speed and terminal settings for each port. The values used for initialization is taken from the appropriate entry in /etc/ttydefs file (same as /etc/gettydefs), which is the terminal line setting table.
- 2.It writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the Break key, the ttymon monitor tries the next speed and writes the prompt again.
- 3.When valid input is received, the ttymon monitor creates a /var/adm/utmp entry, establishes the service environment, and then invokes the login service associated with the part.
- 4.After the service terminates, the ttymon monitor cleans up the /var/adm/utmp entry (if one exists), and returns the port to its initial state.

Note: /var/adm/utmp lists currently logged in users.

#### **[Using Autobaud]**

The ttymon monitor tries to determine the baud rate on the port automatically if autobaud is enabled for a port. Users must press return before the ttymon monitor can recognize the baud rate and print the prompt. Currently the baud rates that can be determined by autobaud are 110 - 9600.

#### **[Bi-directional Service]**

If a port is configured for bi-directional service, the ttymon monitor enables users to connect to a service and, if the port is free, enables communications commands, such as tip for dialing out.

#### **::sacadm**

The /usr/sbin/sacadm command is used to add, remove, enable, disable, and list the ttymon and listen port monitor tables (\_pmtab) using port monitor tags (directories).

Other sacadm functions include listing the current status of port monitors and changing their operational states.

#### **::pmdm**

The /usr/sbin/pmdm command is used to add, remove, enable, disable, and list service tags.

The combination of service tag and port monitor tag uniquely identifies a service instance.

#### **::ttyadm**

The /usr/sbin/ttyadm command creates the serial port service tag when used with the pmdm command.

#### b24.12 **::sacadm**

The sacadm command is used to add the ttymon port monitor tag. This procedure associates the port monitor ttymon with the port monitor tag.

```
=> sacadm -a -p pmtag -t ttymon -c command -v version
```

#### Options

- a Specifies the add option to add a port monitor.
- p Specifies the pmtag (name variable zsmom used here) associated with the port monitor being added.
- t Specifies the port monitor type (ttymon or listen).
- c Identifies the command string to start the port monitor.
- v Specifies the port monitor version number.

#### **[To add port monitor tags]**

```
> sacadm -a -p newpmtag -t ttymon -c /usr/lib/saf/ttymon -v 1
```

The version number is found using /usr/sbin/ttyadm -V.

#### **::pmdm**

Use the pmdm command to add services to a port monitor table. The example below adds a login service for an ASCII terminal from a third serial port using an ALM (asynchronous line multiplexer) card.

```
=> pmdm -a -p pmtag -s service_tag -i identity -fflag -v version -m "ttyadm -l tty_label -d device -T term_type -i 'message' -s service -S y|n"
```

## Options

- a Adds a service for a particular port monitor.
- p Specifies the pmtag associated with the port monitor.
- s Specifies the service tag (name variable ttya used).
- i Specifies the identity to be assigned to the service.
- f Specifies a flag associated with the service to be added: u, to create a utmp entry; or x, to not enable the service.
- v Specifies the port monitor version number.
- m Specifies ttymon-specific configuration information provided by the ttyadm command.
- r Specifies to remove service for a port.

### B24.14 [To add services to a serial port monitor table]

```
> pmadm -a -p newpmtag -s ttyc -i root -fu -v l -m ""ttyadm -l 9600 -d /dev/term/c -T tvi925 -i 'terminal disabled' -s /usr/bin/login -S y"
```

#### ::ttyadm

The ttyadm command populates the port monitor table (`_pmtab`).

```
=> pmadm -a -p pmtag -s service_tag -i identity -fflag -v version -m ""ttyadm -l tty_label -d device -T term_type -i 'message' -s service -S y"
```

## Options

- l Specifies the baud rate (or ttylabel) from the `/etc/ttydefs` file.
- d Specifies the full path name of the device.
- T Specifies the terminal type.
- i Specifies a message displayed when the service is disabled.
- s Specifies the full path name of the service program.
- S y Turns on software carrier detect.
- m Specifies STREAMS modules to be pushed. (This option is not used in the example below.)

### [To add services to a serial port monitor table]

```
> pmadm -a -p newpmtag -s ttyc -i root -fu -v l -m ""ttyadm -l 9600 -d /dev/term/c -T tvi925 -l 'terminal disabled' -s /usr/bin/login -S y"
```

#### [Display commands]

#### [Listing Port Monitor Status]

The `-l` (list) option can be used to check the current status of all port monitors. If the `-p pmtag` option is used, the output is restricted to one particular port monitor.

```
> sacadm -l [-p pmtag]
```

#### [Listing Port Monitors and Services]

Use the `pmadm` command with the list option to list all port monitors. The `-p pmtag` option may be used (output is restricted).

```
> pmtag -l [-p pmtag]
```

Use the `pmadm` command and options to display specific configured services of a port monitor.

```
> pmadm -l -s ttya
```

### b24.16 [Administration commands]

#### [Killing a Port Monitor]

The `-k` (kill) option can be used to stop a port monitor process.

```
> sacadm -k -p zsmon
```

#### [Starting a Port Monitor]

The `sacadm -s` command can be used to start a killed port monitor.

```
> sacadm -s -p zsmon
```

#### [Disabling a Port Monitor] (future)

Disabling a port monitor prevents new services from being spawned for incoming connections, without interfering with existing services.

```
> sacadm -d -p zsmon
```

#### [Enabling a Port Monitor]

Enabling a port monitor enables it to service new requests.

```
> sacadm -e -p zsmon
```

#### [Removing a Port Monitor Tag]

Removing a port monitor tag deletes all of its associated configuration files. To reconfigure a port monitor tag, remove it and add a new one.

```
> sacadm -r -p zsmon
```

#### [Disabling a Service]

If it is necessary to disable a service, (for example to switch off a terminal login), use the `-d` (disable) option along with the port monitor tag and service tag.

```
> pmadm -d -p zsmon -s ttyb
```

### **[Enabling a Service]**

If a service is disabled, it can be activated using pmadm with the -e (enable) option. It is not possible to enable a service if the port monitor has been killed.

```
> pmadm -e -p zsmon -s ttyb
```

### b24.18 **[::/etc/ttydefs]**

The /etc/ttydefs file defines baud rates and terminal settings for TTY ports. These are used in the ttyadm command.

When ttymon initializes a port, it searches this file for an entry that contains the ttylabel that matches the ttylabel in the port monitor table.

The ttymon port monitor uses this entry as a starting point for setting the proper line-speed settings.

### **[::/usr/share/lib/terminfo]**

Default terminal control settings in the Solaris 2.x environment are found in the /usr/share/lib/terminfo directory, which is a database of device descriptions for terminals and printers.

This database is used by screen-oriented programs such as the vi editor and cursor-based programs.

To check if terminal entries exist in the terminfo database, list the contents of the subdirectories in the /usr/share/lib/terminfo directory. The terminfo entry has a name with the same initial letter or digit as the abbreviation of the terminal. If there is no entry, check if the terminal can emulate any other in the database.

The termcap database and associated utilities are provided in the SunOS/BSD compatibility package, and are intended primarily for application compatibility.

### **[SAF File Summary]**

<b><i>File Name</i></b>	<b><i>Description</i></b>
/etc/saf/_sysconfig	The per-system configuration script.
/etc/saf/_sactab	The sac commands' administrative file that contains configuration data for the port monitor's sac controls.
/etc/saf/pmtag	The home directory for pmtag port monitor.
/etc/saf/pmtag/_pmtab	The pmtag's administrative file that contains port monitor-specific configuration data for the services pmtag provides.
/var/saf/_log	The sac commands' log file. (Remove occasionally)
/var/saf/pmtag	The directory for the pmtag log files. (Remove occasionally)