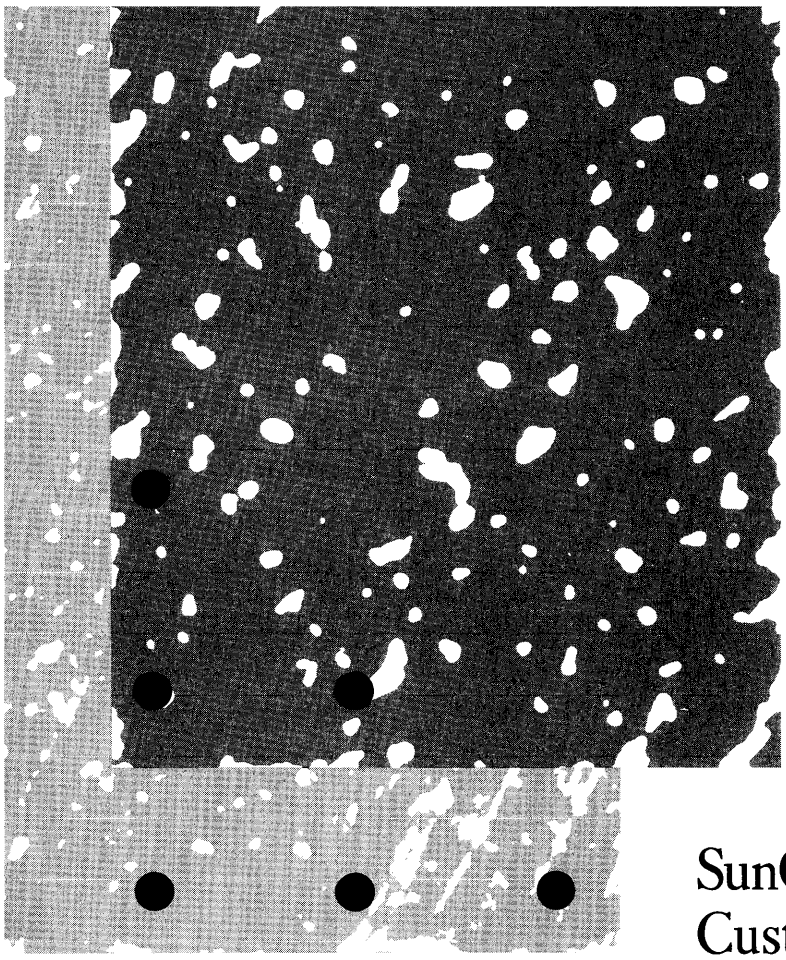


SunOS User's Guide: Customizing Your Environment

Contents

Figures

Preface

SunOS User's Guide: Customizing Your Environment describes the setup files for the C shell command interpreter and the interactive editor `vi`. (See *SunOS User's Guide: Getting Started* for tutorials on these topics.) Each of these files is read in automatically by the appropriate program, and contains commands and instructions to set up (or disable) various features of that program.

Your *environment* is— loosely defined— the various options and features that affect how the system or interactive program responds to you.

There is a sample of each setup file and a line-by-line explanation of its contents. Culled from a variety of expert users, these files contain some very convenient combinations of features and commands. Most importantly, these samples provide a starting point from which you can begin tailoring the system to your specific needs and style.

Chapter 1 is an overview of various setup files and a description of how they are used by the interactive programs.

Chapter 2 describes the `.cshrc` file for the C shell.

Chapter 3 describes the `.login` file for the C shell.

Chapter 4 describes the `.logout` file for the C shell.

Chapter 5 describes the `.exrc` file for `vi` (and the line editor `ex`).

Online copies of the sample files are located in:

```
.cshrc      /usr/lib/Cshrc
.login      /usr/lib/Login
.logout     /usr/lib/Logout
.exrc       /usr/lib/Exrc
```

Other chapters describe how to mount file systems from remote machines with NFS and RFS, and how to use the automounter.

1

Overview

If you have been using the system for a while, you have probably discovered features that you like and features you *would* like. The interactive programs that you have used so far have many optional features that you may not know about. This manual describes a number of these features and how to set things up so that you get the features you want automatically.

1.1. The SunOS Environment

When working with the system, the interactive program that is currently running on your terminal provides a context in which you do your work. When you first log in, you are said to be "in" the command interpreter or *shell*. When you change directories, you are said to be "in" a new one. When using the text editor, you are said to be "in" `vi`.

While in the shell, you typically run the commands described in the *SunOS Reference Manual*. When in `vi`, you typically use text editing commands to read and modify files, as described in *SunOS User's Guide: Getting Started*. While in `mail` or Mail Tool, you typically use commands to read and dispose of messages, or to compose and post messages.

In keeping with this analogy, you can think of the environment as characteristics of the system and the current interactive program that affect the way you work.

When you change interactive programs (by running `mail` for instance), some characteristics, such as the commands that are accepted, also change. Others, such as the current working directory, may not. But most importantly, you usually wish to perform different sorts of tasks, so your expectations about what is a proper response from the system also change.

The technical meaning of *environment* with respect to the SunOS operating system is more restricted: the environment is a body of information that is inherited from the "parent" of a process (program currently running). For example, the name of the current directory is passed along when you start `vi`, so it appears that you "stay in the same directory." For more information about processes, see *SunOS User's Guide: Doing More.*

1.2. Interactive Programs and Setup Files

The interactive programs that you use most often, such as the C shell command interpreter, `mail`, and `vi`, have a variety of optional settings that affect the way they respond to your commands. Unlike options that you type in on the command line (such as `ls -l`), you typically select interactive features by typing in commands while you are using that program.[1]

[1] Most interactive commands also have command-line options that you can specify. Refer to the entry for the command of interest in the *SunOS Reference Manual* for information about its command-line options.

To save you time, most interactive programs allow you to put a list of commands (to select features that you normally want in effect) in a *setup* file in your home directory. Each program reads its setup file(s) automatically and performs the commands it contains.

Subsequent chapters present samples of the various setup files for `vi` and the C shell command interpreter. Each file is described command by command.

The `.mailrc` File

`.mailrc`, a file that modifies the behavior of `mail` and Mail Tool, is discussed in *SunOS User's Guide: Getting Started*.

1.3. Installing the Setup Files in Your Home Directory

The sample files described in this manual are located in the directory `/usr/lib`, and you can copy them to your home directory. But before you do, check to see if there are setup files already present:

```
venus% cd
venus% ls .cshrc .exrc .login .logout
.cshrc not found
.exrc not found
.login not found
.logout not found
venus%
```

CAUTION **If one or more of these files *are* present in your home directory, check with your system administrator before you copy the samples.**

To copy the setup files, type in these commands:

```
venus% cd
venus% cp /usr/lib/Cshrc  .cshrc
venus% cp /usr/lib/Exrc   .exrc
venus% cp /usr/lib/Login  .login
venus% cp /usr/lib/Logout .logout
```

Once copied, you can modify them as you like using `vi` or any other text editor.

These samples have been culled from the setup files of a variety of expert users. They contain many useful features and ideas. Even so, you will want to edit them to suit your own personal needs and tastes, and to remove references to features that you don't want. Generally speaking, the smaller the setup file is, the faster the program will start up.

1.4. Setting Up Your Terminal

Underlying all of these considerations is the terminal you are using and *its* characteristics. As mentioned in *SunOS User's Guide: Getting Started*, you can assign terminal functions such as `erase` and `kill` (erase the entire line typed in so far) to control keys such as Del and Back Space.[2]

[2] If you are using SunView on the Sun workstation, you can assign commands and functions to the special function keys on the keyboard. See *SunView User's Guide* for details.

The command `stty` provides you with a means to set up these and other terminal characteristics. To find out what your current terminal characteristics are, type in the command:

```
stty everything
```

Or

```
stty -a
```

This gives you a list of all terminal characteristics currently in effect:

```
venus% stty everything
speed 9600 baud, 34 rows, 80 columns
parenb -parodd cs7 -cstopb -hupcl cread -clocal -crtscts
-ignbrk brkint ignpar -parmrk -inpck istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff imaxbel
isig iexten icanon -xcase echo echoe echok -echonl -noflsh -tostop
echoctl -echoprt echoke
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
erase  kill   werase rprnt  flush  lnext  susp   intr   quit  stop   eof
^?     ^U     ^W     ^R     ^O     ^V     ^Z/^Y  ^C     ^\    ^S/^Q  ^D
```

The command

```
stty all
```

displays a shorter list. For now, you can ignore all but the last two lines,[3] which describe terminal-control functions and the keys they are set to. These are described below:

```
venus% stty all
speed 9600 baud, 34 rows, 80 columns; evenp
-inpck imaxbel
iexten crt
erase  kill   werase rprnt  flush  lnext  susp   intr  quit  stop   eof
^?     ^U     ^W     ^R     ^O     ^V     ^Z/^Y  ^C    ^\    ^S/^Q  ^D
```

`erase`
: *Erase character.* Backspace and erase one character. This is the Del key by default on some keyboards. On others, this function is assigned to the Back Space key.

`kill`
: *Kill the whole line.* Erase the entire command line typed in so far.

`werase`
: *Delete word.* Erase the last word typed in (back to a space or tab); usually assigned to Ctrl-W.

[3] For more information about the remaining terminal characteristics, refer to `stty` in the *SunOS Reference Manual*.

`rprnt`
Reprint. Reprint the line typed in so far. This is useful when you type ahead and the prompt gets displayed in the middle of your text.

`flush`
Wait for a keystroke. Stops terminal output until you press a key.

`lnext`
Literal next-character. Interprets the next (control) character as literal text. This is useful for entering control or escape characters.

`susp`
Suspend the program. Temporarily halts execution of the program currently running and puts it in the background. To resume execution of the command, type `%`.[4] When you type the suspend character, usually Ctrl-Z, in the middle of a C shell command-line, the shell ignores that line and issues a new prompt.

`intr`
Interrupt. Interrupt the program currently running.

`quit`
Halt the current program and leave a binary image in a file called `core`.

`stop`
Stop the display. To resume, press Ctrl-Q.

`eof`
End-of-file. Send the program an end-of-file character.

To assign any of these functions to another control key, supply the function, a space, and the circumflex character (`^`), followed by the new key, as an arguments to `stty`:

A question mark has a special meaning to the SunOS operating system; you must *escape* it with a backslash to make it a simple ? character.

```
venus% stty werase ^H
venus% stty erase ^\?
venus%
```

This avoids problems trying to type in a key that is already assigned.

To assign `erase` to the Back Space key, use the command:

```
stty erase ^h
```

To assign `werase` to the Del key, use:

```
stty werase ^\?
```

To assign `kill` to the Esc key, use:

[4] C shell only.

```
stty kill ^[
```

Chapter 3 has more information about setting up terminal characteristics. For a description of other terminal characteristics that you can set up, refer to `stty` in the *SunOS Reference Manual.* To assign commands to the special function keys on the workstation keyboard, refer to *SunView User's Guide.*

1.5. The File System Hierarchy

The file system's directory hierarchy is a part of the "landscape" that you will want to become familiar with. To see an example of a typical SunOS hierarchy, type **`man hier`** or look under *hier* in Section 7 of the *SunOS Reference Manual.*

2

The C Shell and the .cshrc File

The C shell is one of the two command interpreters available on the Sun workstation, and it's the one that we recommend for interactive use. Whenever you start running the C shell, such as when you log in or open a `shelltool` or `cmdtool`, the C shell looks for the `.cshrc` file in your home directory for its initial instructions. You can include in this file any command that you might ordinarily type on the command line.

The name is derived from `csh`, which is the program that uses it. Setup files ending in `rc` are read at the time you run the *command*. The dot at the beginning of the filename indicates that this file is to remain hidden from view when you give the `ls` command. Setup files are rarely of interest unless you are editing them specifically, and in that case you already know the filename and directory location. (To list hidden files, use the `-a` option of `ls`.)

2.1. Selecting C Shell Features

While in the C shell, you can use the `set` command to select the options you would like. For instance, if you want the C shell to prevent you from accidentally logging out by typing a Ctrl-D, you can set the `ignoreeof` option (or, technically speaking, *variable*):

```
venus% set ignoreeof
venus% ^D
Use "exit" to leave csh.
```

To turn off an option, use the `unset` command:

```
venus% unset ignoreeof
```

Some options allow you to supply a specific number or value. For instance, you can use the `history` variable to select the size of the history list; that is, the number of previous commands to remember:

```
venus% set history=40
```

Or you can alter the prompt that the shell displays:

```
venus% set prompt = "THIS IS A VERY LONG PROMPT: "
THIS IS A VERY LONG PROMPT: ■
```

To see what options are currently in effect, and their values (if any) use the `set` command with no arguments:

```
venus% set
argv    ()
cdpath  (. .. /home/sam /home/sam/bin /usr/sam/src)
cwd     /home/sam/env
history 40
home    /home/sam
ignoreeof
noclobber
notify
path    (. /home/sam /home/sam/bin /usr/ucb /usr/bin /bin)
prompt  venus%
shell   /bin/csh
status  0
term    sun
user    sam
```

You can find descriptions of all C shell options (predefined variables) under `csh` in the *SunOS Reference Manual.* (Or type `man csh`.)

2.2. A Sample `.cshrc` File

The following pages contain an annotated listing of the sample `.cshrc` file located in `/usr/lib/Cshrc`. This is a very large sample file. Many of the commands and features it includes may not pertain to you, and we recommend that you delete those that you don't want from your copy of the file.

A number of commands have been "commented out" by placing a pound sign (#) to their left. The C shell will ignore these commands unless you remove the pound sign character. Commands that are listed but commented out are felt to be interesting and educational, but not necessarily those that a beginner would use.

```
#################################################################
#
#          .cshrc file
#
#          initial setups for both interactive and noninteractive
#          C-Shells
#
#################################################################


#          set up search path

set lpath = ( )  #  add directories for local commands                    1
set path = (. ~ ~/bin $lpath /usr/local /usr/ucb /usr/bin /bin)           2

#          cd path

set lcd = ( )  #  add parents of frequently used directories             3
set cdpath = (.. ~ ~/bin ~/src $lcd)                                      4

#          set this for all shells

set noclobber                                                             5

#          aliases for all shells

alias cd              'cd \!*;echo $cwd'                                  6
alias cp              'cp -i'                                             7
alias mv              'mv -i'                                             8
alias rm              'rm -i'                                             9
alias pwd             'echo $cwd'                                         10
#alias del            'rm -i'                                            11
#umask 002                                                               12

#          skip remaining setup if not an interactive shell

if ($?USER == 0 || $?prompt == 0) exit                                    13

#           settings  for interactive shells

set history=40                                                            14
set ignoreeof                                                             15
#set notify                                                               16
#set savehist=40                                                          17
#set prompt="% "                                                          18
#set prompt="`hostname`{`whoami`}\!: "                                    19
#set time=100                                                             20

#           commands for interactive shells
```

```
#date                                                                    21
#pwd                                                                     22

#          other aliases

#alias a              alias                                              23
#alias h              'history \!* | head -39 | more'                    24
#alias u              unalias                                            25

#alias ^L             clear                                              26
#alias list           cat                                                27
#alias lock           lockscreen                                         28
#alias m              more                                               29
#alias mroe           more                                               30
#alias type           more                                               31

#alias .              'echo $cwd'                                        32
#alias ..             'set dot=$cwd;cd ..'                               33
#alias ,              'cd $dot '                                         34

#alias dir            ls                                                 35
#alias pdw            'echo $cwd'                                        36
#alias la             'ls -a'                                            37
#alias ll             'ls -la'                                           38
#alias ls             'ls -F'                                            39

#alias pd             dirs                                               40
#alias po             popd                                               41
#alias pp             pushd                                              42

#alias +w             'chmod go+w'                                       43
#alias -w             'chmod go-w'                                       44
#alias x              'chmod +x'                                         45

#alias j              'jobs -l'                                          46

#alias bye            logout                                             47
#alias ciao           logout                                             48
#alias adios          logout                                             49

#alias psg            'ps -ax | grep \!* | grep -v grep'                 50
#alias punt           kill                                               51

#alias r              rlogin                                             52
#alias run            source                                             53

#alias nms 'tbl \!* | nroff -ms | more'                 # nroff -ms      54
#alias tms 'tbl \!* | troff -t -ms >! troff.output &'   # troff -ms      55
#alias tpr 'tbl \!* | troff -t -ms | lpr -t &'          # troff & print  56
```

```
#alias ppr 'lpr -t \!* &'                                    # print troffed    57

#alias lp1          'lpr -P1'                                                    58
#alias lq1          'lpq -P1'                                                    59
#alias lr1          'lprm -P1'                                                   60

#alias sd           'screendump | rastrepl | lpr -v &'                           61

#alias edit         textedit                                                     62

#alias help         man                                                          63
#alias key          'man -k'                                                     64

#alias mkae         make                                                         65
```

Explanation of Command Lines

Line 1:

```
set lpath = ( )  #  add directories for local commands
```

Creates a variable in which to add the pathnames of directories containing local commands. Add the pathnames for any such directories between the parentheses (ask your system administrator for the appropriate names). These directories are incorporated into the `path` variable in line *2*.

Line 2:

```
set path = (. ~ ~/bin $lpath /usr/local /usr/ucb /usr/bin /bin
```

Sets the `path` variable to include the standard directories containing SunOS commands.

Line 3:

```
set lcd = ( )  #  add parents of frequently used directories
```

Creates a variable in which to add the pathnames of directories that are *parents* of those that you often `cd` to. For instance, if you often `cd` to `/usr/man/man1`, put the pathname `/usr/man` between the parentheses. These directories are added to the `cdpath` variable in the next line.

Line 4:

```
set cdpath = (.. ~ ~/bin ~/src $lcd)
```

Sets the `cdpath` variable. You need not specify pathnames when you `cd` to directories that are contained in any of those listed. With `..` set in your `cdpath` (as above), if you were in `/usr/man/man1` and you wanted to `cd` to `/usr/man/cat1`, you could use the command `cd cat1` to do so. Note that `cdpath` is followed linearly, so if both `~/bin` and `~/src` contain a subdirectory called `local`, and you enter the command `cd local`, you will change directory to `~/bin/local`.

Line 5:

```
set noclobber
```

Prevents unintentional overwrite of files when copying `cp`)(a file onto an already existing file. Also prevents unintentional overwrites of files when you use the > symbol. See *SunOS User's Guide: Doing More* for details.

Line 6:

```
alias cd        'cd \!*;echo $cwd'
```

Displays the new directory when you use `cd`. to change directories. See *SunOS User's Guide: Getting Started* for more on aliases.

Line 7:

```
alias cp        'cp -i'
```

Asks for confirmation before overwriting existing files with cp.

Line 8:

```
alias mv        'mv -i'
```

Asks for confirmation before overwriting existing files with mv.

Line 9:

```
alias rm        'rm -i'
```

Asks for confirmation before removing files.

Line 10:

```
alias pwd       'echo $cwd'
```

A faster way of finding out which directory you're in than the built-in pwd command.

Line 11:

```
#alias del       'rm -i'
```

A name for rm that is familiar to PC users.

Line 12:

```
#umask 002
```

Sets the default permissions mask for new files to allow read and write access to the owner's group as well as to the owner.

Line 13:

```
if ($?USER == 0 || $?prompt == 0) exit
```

Tests to see whether there is a variable called USER, or a variable called prompt currently set. If neither of them is not set, then the shell is non-interactive, so the C shell doesn't bother to process the rest of the commands from this file. This saves time running subshells.

Line 14:

```
set history=40
```

The C shell records the last 40 commands typed in.

Line 15:

```
set ignoreeof
```

Prevents accidental logouts when you type Ctrl-D.

Line 16:

```
#set notify
```

Prevents waiting for display of C shell messages. Normally, the C shell waits until just before printing the prompt to print its messages. Commands, however, don't always wait to print their messages, so setting `notify` means that all messages will work the same way.

Line 17:

```
#set savehist=40
```

When you log out, the C shell saves the last 40 commands, and uses them as the starting history list for your next session.

Line 18:

```
#set prompt="% "
```

An alternate prompt favored by some SunOS wizards.

Line 19:

```
#set prompt="'hostname'{'whoami'}\!: "
```

An alternate prompt favored by some network wizards. This prompt gives the name of the machine, followed by the name of the user, followed by the history number of the command:

```
venus{medici}22:
```

Line 20:

```
#set time=100
```

Display time statistics for commands that take longer than 100 CPU seconds.

Line 21:

```
#date
```

Display the date and time when the C shell starts up.

Line 22:

```
#pwd
```

Display the working directory when the C shell starts up.

Line 23:

```
alias a         alias
```

Abbreviate the `alias` command.

Line 24:

```
#alias h        'history \!* | head -39 | more'
```

Abbreviate `history`, and delete the last line (containing `h`) from the display. (Assumes that you have the `history` variable set to 40, as above.)

Line 25:

```
#alias u        unalias
```

Abbreviate the `unalias` command.

Line 26:

```
#alias ^L       clear
```

The Ctrl-L character is the character often used to begin a new page or clear the current one. This alias mimics that behavior.

Line 27:

```
#alias list     cat
```

A name for `cat` that is familiar to PC users.

Line 28:

```
#alias lock     lockscreen
```

An abbreviation for `lockscreen`.

Line 29:

```
#alias m        more
```

An abbreviation for `more`.

Line 30:

```
#alias mroe     more
```

A remedy for "fat fingers" (ie., for mistyping the word "more").

Line 31:

```
#alias type     more
```

A name for `more` that is familiar to PC users.

Line 32:

```
#alias .        'echo $cwd'
```

An abbreviation for `pwd`.

Line 33:

```
#alias ..       'set dot=$cwd;cd ..'
```

A quick way to change from child directory to parent (and back again with alias on the next line).

Line 34:

```
#alias ,          'cd $dot '
```

A quick way to change back after using the `..` alias above.

Line 35:

```
#alias dir        ls
```

A name for `ls` that is familiar to PC users.

Line 36:

```
#alias pdw        'echo $cwd'
```

For those people who often mistype "pwd." Has the same effect as the `.` alias for `pwd`, above.

Line 37:

```
#alias la         'ls -a'
```

Abbreviation for command to list all filenames, including those that begin with a dot (`.`).

Line 38:

```
#alias ll         'ls -la'
```

Abbreviation for a command to give a long listing of filenames, including those that begin with a dot.

Line 39:

```
#alias ls        'ls -F'
```

This way, `ls` appends characters on the end of a filename to indicate that file's type. These characters are / (slash) for directories; * for executable files; `@` for symbolic links; and = for AF_UNIX domain sockets.

Line 40:

```
#alias pd         dirs
```

Abbreviation to display the directory stack maintained by `pushd` and `popd`. See *SunOS User's Guide: Getting Started* for details.

Line 41:

```
#alias po         popd
```

Change directories to the one on the top of the stack, and remove its name from the stack. See *SunOS User's Guide: Getting Started* .

Line 42:

```
#alias pp        pushd
```

Change directories, adding the current directory and the destination to the stack. See *SunOS User's Guide: Getting Started*.

Line 43:

```
#alias +w       'chmod go+w'
```

Make a file writable to the group and public.

Line 44:

```
#alias -w       'chmod go-w'
```

Make a file unwritable to all but you, the owner.

Line 45:

```
#alias x         'chmod +x'
```

Give a file execute permissions for all users.

Line 46:

```
#alias j         'jobs -l'
```

Display the list of background jobs. With `-l`, the `j` alias also gives the process id number of the stopped job.

Line 47:

```
#alias bye       logout
```

Another name for `logout`.

Line 48:

```
#alias ciao     logout
```

An international term for `logout`.

Line 49:

```
#alias adios     logout
```

The traditional South-of-the-Border way to log out.

Line 50:

```
#alias psg       'ps -ax | grep * | grep -v grep'
```

Check on the status of a command by its name. See *SunOS User's Guide: Doing More* for details.

Line 51:

```
#alias punt      kill
```

Another name for `kill`.

Line 52:

```
#alias r         rlogin
```

Log in to another host machine on the net. See *SunOS User's Guide: Getting Started* for details.

Line 53:

```
#alias run       source
```

The `source` command instructs the C shell to take a file (such as the `.cshrc` file) as a list of commands to perform.

Line 54:

```
#alias nms 'tbl \!* | nroff -ms | more'
```

Format and display a document containing `tbl` instructions and `ms` macros on the terminal. For example, to format the file `wombat`, you would type `nms wombat`.

Line 55:

```
#alias tms 'tbl \!* | troff -t -ms >! troff.output &'
```

Format a document using `tbl` and `ms` macros, and place the output in a file for later printing.

Line 56:

```
#alias tpr 'tbl \!* | troff -t -ms | lpr -t &'
```

Format a document using `tbl` and `ms` macros and print it.

Line 57:

```
#alias ppr 'lpr -t \!* &'
```

Print a preformatted troff-output file.

Line 58:

```
#alias lp1       'lpr -P1'
```

Abbreviation to print on printer #1. See *SunOS User's Guide: Getting Started* for details.

Line 59:

```
#alias lq1       'lpq -P1'
```

Abbreviation to check the queue for printer #1.

Line 60:

```
#alias lr1        'lprm -P1'
```

Abbreviation to remove a job or jobs from printer #1.

Line 61:

```
#alias sd         'screendump | rastrepl | lpr -v &'
```

Abbreviation to print the contents of the workstation screen.

Line 62:

```
#alias edit       textedit
```

Abbreviation for the window-system text editor.

Line 63:

```
#alias help       man
```

Another name for the `man` command.

Line 64:

```
#alias key        'man -k'
```

Abbreviation for the `man -k` command (same as the `whatis` command described in *SunOS User's Guide: Doing More*).

Line 65:

```
#alias mkae       make
```

Useful if you tend to misspell the word "make."

3

The C Shell and the .login File

When you log in, after performing instructions in the .cshrc file, the C shell then performs instructions in the .login file. Subsequent C shells, such as the shelltool or cmdtool windows, ignore the .login file.

Like the .cshrc file, you can include any command that you might type in on the command line. However, we recommend that you use the .login file for initializing remote terminals (for when you log in by phone), starting your window system (when you first log in to the workstation), and setting up special variables called *environment* variables. Unlike shell variables and aliases, environment variables are passed along to subsequent commands and programs automatically. You need not set them up again every time you start a new C shell or run a new program such as vi.

Environment variables are useful for storing information that all programs need to know about. For instance, many commands and programs need to know what type of terminal you are using. This information is stored in the TERM environment variable. Commands that send output to the printer need to know which printer to send their output to. You can use the PRINTER environment variable, to store the name of a printer to use by default.

To set an environment variable, use the setenv command. This command has two required arguments, the *name* of the variable, and its *value*:

```
setenv name value
```

For example:

```
venus% setenv PRINTER lw
```

Although not required, the convention is to use all capitals for names of environment variables (to distinguish them from ordinary shell variables). To see what environment variables are currently in effect, use the printenv command:

```
venus% printenv
HOME=/home/sam
SHELL=/bin/csh
PATH=.:/home/sam:/home/sam/bin:/usr/local:/usr/ucb:/usr/bin:/bin
TERM=sun
USER=sam
EDITOR=/usr/ucb/vi
PRINTER=lw
WINDOW_PARENT=/dev/win0
WMGR_ENV_PLACEHOLDER=/dev/win1
WINDOW_ME=/dev/win9
WINDOW_GFX=/dev/win9
```

To remove an environment variable, use the `unsetenv` command:

```
venus% unsetenv PRINTER
venus% echo $PRINTER
PRINTER: Undefined variable.
```

3.1. A Sample `.login` File

The following pages contain an annotated listing of the sample `.login` file located in `/usr/lib/Login`. If you do not plan to log in from a remote terminal over the phone, you can delete the lines that pertain to remote terminals. Again, some commands are commented out. And again, we recommend that you delete or comment out commands that do not pertain to you.

```
#################################################################
#
#          .login file
#
#          Read in after the .cshrc file when you log in.
#          Not read in for subsequent shells.  For setting up
#          terminal and global environment characteristics.
#
#################################################################


#          terminal characteristics for remote terminals:

#          Leave lines for all but your remote terminal commented
#          out (or add a new line if your terminal does not appear).
if ($TERM != "sun") then                                                    1
#eval `tset -sQ -m dialup:?925 -m switch:?925 -m dumb:?925 $TERM`           2
#eval `tset -sQ -m dialup:?h19 -m switch:?h19 -m dumb:?h19 $TERM`           3
#eval `tset -sQ -m dialup:?mac -m switch:?mac -m dumb:?mac $TERM`           4
#eval `tset -sQ -m dialup:?vt100 -m switch:?vt100 -m dumb:?vt100 $TERM`     5
#eval `tset -sQ -m dialup:?wyse-nk -m switch:?wyse-nk -m dumb:?wyse-nk $TERM` 6
#eval `tset -sQ -m dialup:?wyse-vp -m switch:?wyse-vp -m dumb:?wyse-vp $TERM` 7
endif                                                                       8

#          general terminal characteristics

#stty -crterase                                                             9
#stty -tabs                                                                10
#stty crt                                                                  11
#stty erase '^h'                                                           12
#stty werase '^?'                                                          13
#stty kill '^['                                                            14
#stty new                                                                  15

#          environment variables

#setenv EXINIT 'set sh=/bin/csh sw=4 ai report=2'                          16
setenv MORE '-c'                                                           17
#setenv PRINTER lw                                                         18

#          commands to perform at login

#w                                                                         19

if ("`tty`" != "/dev/console") exit                                        20
echo -n "SunView? (^C to interrupt) "                                      21
sleep 5                                                                    22
sunview                                                                    23
```

Explanation of Command Lines

Line 1:

```
if ($TERM != "sun") then
```

Perform the commands between this line and the `endif` line only when logging in on a terminal, rather than a Sun workstation.

Line 2:

```
#eval 'tset -sQ -m dialup:?925 -m switch:?925 -m dumb:?925 $TERM'
```

If logging in over a phone line, or some other remote means, set up terminal characteristics for a Televideo 925 terminal and place these characteristics in the environment for faster startup of interactive programs. Asks if terminal type 925 is correct by prompting you with

```
TERM 925
```

Pressing Return accepts the 925 as the terminal type; or you can enter another terminal type (from `/etc/termcap`). Refer to `tset` in the *SunOS Reference Manual* for more information.

All of the lines pertaining to specific types are commented out. To activate the line that pertains to your terminal, remove the pound-sign. If your terminal does not appear, duplicate this line, change the `925` to the name of your terminal (see your system administrator for this information) and remove the pound sign.

Line 3:

```
#eval 'tset -sQ -m dialup:?h19 -m switch:?h19 -m dumb:?h19 $TERM'
```

Set up terminal characteristics for a Heathkit H19 terminal.

Line 4:

```
#eval 'tset -sQ -m dialup:?mac -m switch:?mac -m dumb:?mac $TERM'
```

Set up terminal characteristics for a Macintosh running Macterminal.

Line 5:

```
#eval 'tset -sQ -m dialup:?vt100 -m switch:?vt100 -m dumb:?vt100 $TERM'
```

Set up terminal characteristics for a VT100 terminal.

Line 6:

```
#eval 'tset -sQ -m dialup:?wyse-nk -m switch:?wyse-nk -m dumb:?wyse-nk $TERM'
```

Set up terminal characteristics for a Wyse 50 terminal.

Line 7:

```
#eval 'tset -sQ -m dialup:?wyse-vp -m switch:?wyse-vp -m dumb:?wyse-vp $TERM'
```

Set up terminal characteristics for a Wyse 50 in ADDS Viewpoint mode with "enhance" turned on.

Line 8:

```
endif
```

Marks last line to be skipped when an `if ... then` statement is found to be false; in this case, when logging in to a Sun workstation directly (or from another Sun on the network).

Line 9:

```
#stty -crterase
```

Set up the erase function to backspace without blotting out erased characters. Erased characters remain visible on the screen until you overwrite them with new ones, but are not transmitted to the C shell when you press Return.

Line 10:

```
#stty -tabs
```

Convert tabs to spaces when displayed on the screen.

Line 11:

```
#stty crt
```

Set up standard CRT characteristics.

Line 12:

```
#stty erase '^h'
```

Set the erase character to Back Space. Note that with `stty`, control characters are indicated by the two-character symbol *circumflex-character*: `^c`.

Line 13:

```
#stty werase '^?'
```

Set the erase-word character to Del.

Line 14:

```
#stty kill '^['
```

Set line-kill character to Esc. Do not use this option if you use the `vi` editor. `vi` interprets an Esc as "change from insert mode to command mode."

Line 15:

```
#stty new
```

Use the new version of the terminal driver.

Line 16:

```
#setenv EXINIT 'set sh=/bin/csh sw=4 ai report=2'
```

Another way to set up options for `vi`. `vi` recognizes the EXINIT variable as a startup command. If this variable is undefined, `vi` checks for startup commands in `~/.exrc`. However, if an `exrc` file exists in the current directory, `vi` takes its startup commands from this file, ignoring both `~/.exrc` and EXINIT.

Line 17:

```
setenv MORE '-c'
```

Sets up `more` to overwrite the screen, rather than scrolling.

Line 18:

```
#setenv PRINTER lw
```

Indicate which printer is to receive jobs by default.

Line 19:

```
#w                    # see who is logged in
```

See who is logged in on your system.

Line 20:

```
if ("`tty`" != "/dev/console") exit
```

If the terminal is not your workstation console (the workstation when not running the window system), then stop further processing of this file.

Line 21:

```
echo -n "Sunview? (^C to interrupt) "
```

Warn you that SunView is about to start.

Line 22:

```
sleep 5
```

Wait 5 seconds before starting SunView, to give you a chance to press Ctrl-C.

Line 23:

```
sunview
```

Start the SunView window system.

4

The C Shell and the `.logout` File

When you log out completely (not just from a single window), the C shell performs instructions in the `.logout` file. This file is useful for running housekeeping type commands in the background while you are away.

Like `.cshrc` and `.login` you can include any command that you might type in on the C shell command line. We recommend that you use this file only for displaying information about the session just ending that you want to know about and for running background commands. You should *not* put commands that run interactively in this file, nor should you include commands that take any significant amount of time unless the command runs in the background. Otherwise, someone may be able to interrupt the command and gain unauthorized access to your workstation or terminal.

4.1. A Sample `.logout` File

The following pages contain an annotated listing of the sample `.logout` file located in `/usr/lib/Logout`. Some commands are commented out, and we recommend that you delete commands that do not pertain to you.

```
############################################################
#
#          .logout file
#
#          Read in when you exit from the login shell.
#          For performing housekeeping while your are away.
#
############################################################


clear                                                          1
#echo "`hostname`: `whoami` logged out at `date`"              2
#echo "Goodbye\!"                                             3

#if (-e /usr/games/fortune) /usr/games/fortune -a             4
#if (-r /etc/motd) cat /etc/motd                              5

#unalias rm                                                   6
#nice find ~ '(' -name core -o -name '*.BAK' -o -name '*.CKP' \   7
#          -o -name '#*' -o -name junk ')' \
#          -atime +3 -mtime +3 -user $USER -type f -exec \rm '{}' \; &
```

Explanation of Command Lines

Line 1:

```
clear
```

Clears the terminal screen.

Line 2:

```
#echo "`hostname`: `whoami` logged out at `date`"
```

Displays the name of the host machine, your user name, and the date and time you logged out.

Line 3:

```
#echo "Goodbye\!"
```

A more traditional parting wish.

Line 4:

```
#if (-e /usr/games/fortune) /usr/games/fortune -a
```

If the `fortune` command is available, use it to display one of many humorous sayings.

Line 5:

```
#if (-r /etc/motd) cat /etc/motd
```

If the message of the day is readable, display it.

Line 6:

```
#unalias rm
```

If you used `alias` to change the `rm` command in some way, this resets it to its original, default value.

Line 7:

```
#nice find ~ '(' -name core -o -name '*.BAK' -o -name '*.CKP' \
#       -o -name '#*' -o -name junk ')' \
#       -atime +3 -mtime +3 -user $USER -type f -exec \rm '{}'
```

Run `find` at low priority in the background, starting with your home directory. Look for files named `core`, `*.BAK`, `*.CKP`, `'#*'` or `junk`. Of these, select only those that are at least 3 days old, haven't been modified for at least 3 days, belong to you, and are regular files (not directories). Remove each file selected, escaping any aliases that might be applied to `rm`.

To activate this command, you need to delete the first pound sign in all three lines.

5

vi and the .exrc File

Whenever you run vi, the editor looks in the .exrc file for initial commands and option settings. The vi editor has a number of options that are described in detail in *Editing Text Files*. vi has a :set command with which you select the editing options that you want, but you cannot use it to create new variables, as you can with the C shell's set command.

5.1. Setting Options While in vi

To see the list of options that are currently in effect, type in :set with no arguments:

```
...
~
~
~
~
~
~
~
~
:set
autoindent beautify nomesg number redraw term=sun wrapmargin=8
```

To see the list of current and possible settings, use the :set all command:

```
...
~
:set all
autoindent              open                    tabstop=8
autoprint               nooptimize              taglength=0
noautowrite             prompt                  tagstack
beautify                noreadonly              term=sun
directory=/tmp          redraw                  noterse
noedcompatible          remap                   timeout
noerrorbells            report=5                ttytype=sun
hardtabs=8              scroll=16               warn
noignorecase            shell=/bin/csh          window=33
nolisp                  shiftwidth=8            wrapscan
nolist                  noshowmatch             wrapmargin=8
magic                   noslowopen              nowriteany
nonumber
Hit return to continue
```

To select a specific option or options, include them as arguments to the `:set` command. Note that for options having values, there are *no* spaces between the name, the equal-sign, and the value for that option.

To turn off an option, add the prefix `no` to the name of that option as an argument to `:set`.

```
:set
autoindent beautify nomesg number redraw term=sun wrapmargin=8
```

```
:set noautoindent
:set
beautify nomesg number redraw term=sun wrapmargin=8
```

To change the value of a setting such as `wrapmargin`, use `set` to establish a new value:

```
:set wrapmargin=0
```

(This has the effect of eliminating automatic wrapping at the end of the line).

5.2. A Sample `.exrc` File

The following page contains an annotated listing of the sample `.exrc` file located in `/usr/lib/Exrc`. `vi` does not accept comments in the same way as the C shell: lines in `.exrc` files are commented out with double quotes (") instead of with pound signs (#).

```
"
set autoindent                                                    1
set autoprint                                                     2
set noignorecase                                                  3
set nomesg                                                        4
set noslowopen                                                    5
set noterse                                                       6
set nonumber                                                      7
"
set report=2                                                      8
set tabstop=8                                                     9
set wrapmargin=8                                                  10
"
"map ; :                                                          11
"map g :%                                                         12
"map v ~                                                          13
"map m !} fmt -c                                                  14
"map T !} sort                                                    15
"
```

Explanation of Command Lines

Line 1:

```
set autoindent
```

When adding new lines, maintain the same indentation as the line above.

Line 2:

```
set autoprint
```

Automatically print each line altered within `ex`, the line editor.

Line 3:

```
set noignorecase
```

The case (upper or lower) of a character is significant in searches and substitutions. Use `set ignorecase` to make searches and substitutions case insensitive. But be careful if you do!

Line 4:

```
set nomesg
```

Messages to the terminal do not interfere with the `vi` display.

Line 5:

```
set noslowopen
```

Sets up `vi` for operation with a fast terminal or window. For terminals on slow dialup lines, use `set slowopen` to suspend updates of the screen during insertions for smoother operation.

Line 6:

```
set noterse
```

`vi` gives more complete error messages for beginning users. For shorter messages, use `set terse`.

Line 7:

```
set nonumber
```

Inhibits display of line numbers in both `ex` and `vi`. For a display of line numbers, use `set number`.

Line 8:

```
set report=2
```

Report on all substitutions or deletions that affect more than two lines.

Line 9:

```
set tabstop=8
```

Set tab stops every 8 characters.

Line 10:

```
set wrapmargin=8
```

When a space is typed within 8 characters of the right screen edge, insert a carriage-return at the end of the previous word, starting a new line automatically. To turn off this feature, set `wrapmargin` to 0.

Line 11:

```
map ; :
```

While in `vi` command (*visual*) mode, interpret a semicolon as if you had typed a colon. This allows you to use either a semicolon or a colon as the first character in a substitution command.

Line 12:

```
map g :%
```

While in visual mode, interpret a `g` as if you typed the characters `:%`. This allows you to start commands to substitute throughout the file with either a `g` or a `:%`.

Line 13:

```
map v ~
```

While in visual mode, interpret a `v` as if you typed a `~`, the command to invert the case of a character.

Line 14:

```
map m !} fmt -c
```

While in visual mode, interpret an `m` as if you typed in the command

```
!} fmt -c
```

to adjust line-breaks for the lines between the cursor and the end of the paragraph as close to column 80 as possible (without breaking across words). Refer to *Editing Text Files* and `fmt` in the *SunOS Reference Manual* for more information.

Line 15:

```
map T !} sort
```

While in visual mode, interpret a `T` as if you typed in the command

```
!} sort
```

a command to sort the remaining lines in the paragraph.

6

Login Access to Other Machines

You can log in from your machine to another machine using `rlogin`, described in *SunOS User's Guide: Getting Started*.

Some systems *trust* certain users and certain machines to log in without requiring a password and other security mechanisms.

6.1. Trust Amongst Machines on a Network

You can allow or restrict login access by other users to your machine. You, or your system administrator, can set up your machine so that users must supply a password to log in to your machine, or so that certain users from certain machines simply can't log in to your machine.

When another machine running the UNIX system[5] on your local network *trusts* your username on your machine, that machine doesn't require that you type your password or undergo other security checks to log in. The system files `/etc/passwd` and `/etc/hosts.equiv` control the users and the machines that may log in to your machine and to your account.

`/etc/hosts`

Before we describe the files that control access privileges, let's consider how one machine finds another over the network. The `/etc/hosts` file contains a list of machines — also known as *hosts*— and their corresponding addresses on the network. (The file can also contain any optional nicknames for machines.) Your own machine's address is stored here. Programs use `/etc/hosts` to locate other machines for communicating over the network.

However, if you're running the NIS, the NIS file `hosts` may supplant or supplement your own `/etc/hosts` file; the NIS `hosts` contains the network addresses of every machine on the network.[6] Therefore, you (or your system administrator) need only maintain an up-to-date `/etc/hosts` file when the NIS is not available.

Here is a sample `/etc/hosts` file:

---

[5] The SunOS operating system is an enhanced version of UNIX, a registered trademark of AT&T.

[6] Not every system runs the NIS; not all that do use the `hosts` scheme. See *SunOS User's Guide: Getting Started*.

```
#
# Sun Host Database
#
# If the yellow pages is running, this file is only consulted when booting
#
127.0.0.1       localhost loghost
#
192.9.90.114    gaia
192.9.90.103    venus
```

`/etc/hosts.equiv`

The `/etc/hosts.equiv` file contains a list of machine names that your machine trusts. When a user on one of the machines in the list tries to log in or execute a command on your machine, your machine checks in your `/etc/passwd` file to see whether it should permit access. `/etc/passwd` regulates who can log in on a machine; it contains information about users, including their usernames, their passwords, their ID numbers, and so on.[7]

If that person's username is in `/etc/passwd` or in the NIS password database, then your machine checks `/etc/hosts.equiv`. If it contains the name of the machine that he or she is using, then your machine allows him or her to log in. If the hostname isn't there, then he or she is prompted for a password.

Here is a sample `/etc/hosts.equiv` file that permits any users on machines `gaia`, `pluto`, and `verlaine` (who also appear in your `/etc/passwd` file or in the NIS database) to log in or execute commands on your machine from their machine.

You must be `root` to modify this file. For information on becoming `root`, see *SunOS User's Guide: Getting Started* and *SunOS User's Guide: Doing More.*

```
venus% cat /etc/hosts.equiv
gaia
pluto
verlaine
venus%
```

If your system supports the NIS, you may put a single plus sign (+) all by itself in your `/etc/hosts.equiv` file. This signals that anyone on the network served by the NIS (who is also named in in the password database) may log in to your machine without providing a password. As a matter of fact, `/etc/passwd` also uses the plus sign convention, so you can thus allow everyone on the network access to your machine. You can also selectively bar some people in the NIS database from logging in to your machine.

You can further restrict other people's access to your machine with the `.rhosts` file, located in your home directory. For more on `.rhosts` and `/etc/hosts.equiv`, see the *SunOS Reference Manual*, the *System and Network Administration* manual, or type **`man hosts.equiv`**.

[7] For more information about the `/etc/passwd` file, see *SunOS User's Guide: Doing More* and the *SunOS Reference Manual*, or type an 5 passwd.

7

Mounting Remote Filesystems With NFS

If your workplace is running RFS instead of NFS, then you should refer to Chapter 8. Ask your system administrator which one you are using.

You may very often find that you have a need to access programs or files located on another machine. Many times it is not practical, because of space limitations, to copy the files you need over to your machine. For this reason, NFS (described in Chapter 1) offers you the ability to *mount* other filesystems, allowing you to access files on other machines as though they were on your own.

For example, if your company had a lot of software used by everyone, such as spreadsheets, demos, and text editors, a lot of valuable storage space would be taken up if each user had to have his or her own copies of all the programs. People could use `rlogin` to access other machines, but this would be unsatisfactory if they had to do so every time they wanted to run a program. What your company could do, though, would be to put the relevant files on one host (also known as a *server*) and have users mount the filesystem containing those files.

7.1. The `mount` Command

Suppose your company's demos are located in `/usr/demos` on a server called `slugfest`. You create (or have your system administrator create) a directory, `/usr/demos`, on your own system. This `/usr/demos` is called a *mount point*; you then use the `mount` command to create a sort of invisible connection between `/usr/demos` on `slugfest` and your `/usr/demos` mount point. As long as you maintain the mounting, every time you refer to `/usr/demos`, you access the one on `slugfest` as though it were on your own machine — you can list its contents with `ls`, `cd` in and out of it, view its files with `more`, run its programs, and so on. Mounting allows you to do all of this without having to `rlogin` in to `slugfest`.

Use the `mount` command to mount other filesystems. The general format of the command is

```
mount machine-name:filesystem directory
```

You must be `root` to use `mount`. For information on becoming `root`, see *SunOS User's Guide: Getting Started* and *SunOS User's Guide: Doing More.*

where *machine-name* is the name of the remote machine that contains the filesystem you want to access; *filesystem* is that remote filesystem; and *directory* is a directory on your system. (This directory is the mount point). Here's how you would mount `/usr/demo` and view its contents:

```
venus# mount slugfest:/usr/demo /usr/demo
venus# ls /usr/demo
BUTTONBOX       DIALBOX      Makefile     SUNVIDEO
BUTTONTEST      FACES        SHADER       SUNVIEW
CDROM           GP           SOUND        sunview1
venus#
```

You do not have to mount a remote filesystem to a filesystem of the same pathname. In other words, you don't have to mount `/usr/demo` on `slugfest` to a directory called `/usr/demo` on your machine. You can mount it to any directory you want. In fact, since most users aren't allowed to create directories in `/usr`, you could, for example, create a subdirectory `dem` in your home directory and mount `/usr/demo` there (assume `/home/art/medici` is your home directory):

Use `su` to become `root`.

```
venus% mkdir /home/art/medici/dem
venus% su
Password:       (Type root's password here.)
venus# mount slugfest:/usr/demo /home/art/medici/dem
venus# exit
venus% ls /home/art/medici/dem
BUTTONBOX       DIALBOX      Makefile     SUNVIDEO
BUTTONTEST      FACES        SHADER       SUNVIEW
CDROM           GP           SOUND        sunview1
venus%
```

The important thing is that you make sure that there's a mount point on your system to mount the remote filesystem to. Suppose you try to mount `/usr/demos` to a directory, `/usr/showme`, without creating `/usr/showme` first. You'll get an error like this:

```
venus# mount slugfest:/usr/demos /usr/showme
mount: slugfest:/usr/demos on /usr/showme: no such
file or directory
mount: giving up on /usr/showme
venus#
```

7.2. The `/etc/fstab` File

You must be `root` to modify your `/etc/fstab` file.

Rather than having to use `mount` for every filesystem you want to mount, you can make use of the file `fstab` which is located in the directory `/etc`.[8] `fstab` is a table of filesystems you want to mount, with any filesystem options you want. Every time your system is booted, it checks `fstab` to see what filesystems to mount. This can save you considerable time over using the `mount` command directly for each filesystem you want mounted.

[8] *fstab* stands for "file system table."

The general format of an entry in an `fstab` file is

machine:filesystem directory type option freq pass

where

machine:filesystem
is the remote filesystem you want to mount and the machine on which it is located

directory
is the directory on your system you want to mount it to

type
is the filesystem type, in this case *nfs* or *rfs*

options
are the filesystem options, such as those described in Section 7.3

freq
is the number of days between dumps

pass
partition check pass number

When mounting filesystems through NFS, these last two fields should always be 0. Here's an example of an `/etc/fstab` file:

Figure 7-1 *A Sample* `fstab` *File*

```
dirt:/export/root/venus /              nfs rw                0 0
dirt:/usr               /usr           nfs ro                0 0
dirt:/home/dirt         /home/dirt     nfs rw                0 0
moon:/usr/spool/news    /usr/spool/news nfs ro,soft,bg,intr  0 0
anchor:/usr/games       /usr/games     nfs ro,bg             0 0
boodle:/usr/boodle      /usr/boodle    nfs ro,bg             0 0
boodle:/usr/man         /usr/man       nfs ro,soft,bg,intr   0 0
boodle:/usr/tools       /usr/doctools  nfs rw,hard,bg,intr   0 0
capulet:/usr/demo       /usr/demo      nfs ro,bg             0 0
olympia:/usr/doc        /usr/doc       nfs ro,hard,bg,intr   0 0
olympia:/usr/olympia    /usr/olympia   nfs rw,hard,bg,intr   0 0
buoy:/usr/buoy          /usr/buoy      nfs rw,hard           0 0
yow:/usr/src            /usr/src       nfs,ro,soft,bg,intr   0 0
USR     /mnt        rfs     rw,bg   0 0
```

The `umount` Command

`umount` *unmounts* file systems that you have previously mounted. `umount` works similarly to `mount`; however, you only need to specify the mount point on your system when using it. You do not need to mention the remote machine it's mounted from. So, for example, to unmount `/usr/demos`, which we mounted above, you would type as follows:

```
venus# umount /usr/demos
venus#
```

As with `mount`, you must be `root` to use `umount`.

7.3. Options to `mount` and `umount`

There are various ways to use `mount` and `umount`. A few of the more common ways are given here. Please note that you can combine a number of these options; for the exact syntax, consult the *SunOS Reference Manual* or type **man mount**.

(with no options)
`mount` with no arguments displays what you currently have mounted. You do *not* have to be `root` to use the `mount` command this way.

`-p` ("print") `mount` displays a list of mounted filesystems in the format used in `fstab`. You don't need to be `root` to use `mount` this way, either.

`-v` ("verbose") Displays a message indicating each filesystem in `/etc/fstab` being mounted as it occurs. This option is also available with `umount`.

`-a` ("all") `mount` will attempt to mount all the filesystems in `/etc/fstab`.

filesystem

or

directory
If you use `mount` with either the name of a filesystem to be mounted *or* the name of one of your directories to mount to (but not both), `mount` searches `fstab` for the corresponding entry and mounts the filesystem as indicated there. Also true for `umount`.

`-h` *host*
(This is a `umount` option.) `umount` unmounts all the filesystems which are remote-mounted from the server *host*. In the sample `fstab` file shown above, there are a number of filesystems mounted from the server *boodle*; to unmount them, you would give a command like this:

```
venus# umount -v -h boodle
/usr/man: Unmounted
/usr/boodle: Unmounted
/usr/tools: Unmounted
venus#
```

Filesystem Options

`-o` *options*
There are a number of *options* to `mount`. These affect the access you have to the remote filesystem, and the manner that `mount` goes about trying to mount the system. These are some of the more common filesystem options:

`hard` *or* `soft`
If you soft-mount a remote filesystem, `mount` will try up to a certain

number of times and then give up, returning you an error message; when you hard-mount it, `mount` will continue trying the mount until it is successful.

There are advantages to both hard- and soft-mounting. Hard-mounting ensures that `mount` will not give up on a server that is only temporarily inaccessible, but will eventually connect you. It is a good idea to always hard-mount filesystems that you write to; that way you lessen the chance of data being lost during a write operation. On the other hand, if you are hard-mounted to a server when it crashes, your system may hang up while waiting to connect with the remote server.

This option is available only with NFS.

`bg` *or* `fg`
: You can specify whether you want `mount` to retry failed mounting attempts in the *background* or the *foreground*. When a mount occurs in the foreground, the shell is tied up until the mount succeeds; if you run the mount in the background, the shell and you can do other things during the mount attempt.

`rw` *or* `ro`
: You can mount the other filesystem as *read-write* or *read-only*. It is a good idea to mount filesystems as `ro` if you are only going to access their files and programs, not write or change them.

`intr`
: If you hard-mount a filesystem, `intr` allows you to interrupt the mount from the keyboard (with Ctrl-C).

  This option is only available with NFS.

For information on other options, see the *SunOS Reference Manual.*

Remember, you can combine various options when using the `mount` command — see the *SunOS Reference Manual.*

7.4. The Automounter

Just as the `mount` command removes the need to `rlogin` in to other machines every time you want to access their files, the *automounter* lessens the need for hand-mounting remote systems. The automounter mounts remote file systems "on the fly"; that is, it creates a temporary mount point (in the directory `/tmp_mnt`), mounts the remote system, and unmounts it automatically. It does so without your having to change your `fstab` file; nor do you need to become `root` to mount and unmount remote filesystems.

The mounting takes place invisibly when you refer to the remote filesystem. Thus, you can `cd` to a remote directory, `ls` its contents, `more` a file on another machine, and so on, all without having to mount anything first.

The syntax for using the automounter can vary from site to site, and from machine to machine. Here is a general example of how it is used.[9]

[9] These examples assume your site utilizes the NIS `hosts` and `passwd` databases. (These contain the addresses of machines on the network, and the list of users and their home directories, respectively.) Because

A reference to `/net/`*machine*`/`*filesystem* will cause the automounter to automatically mount *filesystem* from the machine *machine*. For instance, in the example below, the command

```
ls /net/jupiter/var/spool
```

causes the automounter to mount `/var/spool` from the machine `jupiter`; and `ls` lists its contents:

```
venus% ls /net/jupiter/var/spool
cron        lw          ppd         uucppublic
ipd         mail        rwho        vpd
lpd         mqueue      secretmail
lpd.lock    news        uucp
venus%
```

Likewise, a reference to `/homes/`*username* causes the automounter to mount the home directory of the person with the username *username*. Using this syntax, the command

```
cat /homes/bcleaver/rhyme.file
```

mounts the `bcleaver`'s home directory, and `cat` displays the contents of the file `rhyme.file`:

```
venus% cat /homes/bcleaver/rhyme.file
A man, he lived by the sewer
And by the sewer he died
And at the coroner's inquest
They called it sewer-side.
venus%
```

If you go a specified period (usually five minutes) without accessing the remote filesystem, `automount` unmounts it for you. The first time you mount a remote filesystem, the automounter can take a little while; once it's mounted, though—and for as long as it remains mounted—access is more or less immediate.

`automount` is started up automatically when your machine starts ("boots"). However, if your system administrator has not set the NIS maps that it needs, or you have not set local maps for it, the automounter exits (dies) silently.

Use the command

```
ps -aux | grep automount
```

to see if you are running `automount`.

In order to make its mounts, the automounter users a number of *map files* (located in `/etc` and beginning with "auto"); you can modify existing maps or

each site running NFS is unique, consult your system administrator on how to use the automounter. See *SunOS User's Guide: Getting Started* for more information on NIS .

create your own. For example, to use the map `/etc/auto.master`, you would start the automounter as follows:

```
venus% su    (become root)
# automount -f /etc/auto.master
# exit
venus%
```

For more information on using the automounter, see *System and Network Administration*.

7.5. Who's Mounting This System? `showmount`

The `showmount` command gives the opposite information from typing `mount`. It gives information about who has mounted filesystems from a particular server.

For example, let's assume that user `michael` on the machine `yazoo` has mounted `/usr/games` from the server `slugfest`; `reed` on `zippy` has mounted `/usr/demos` from the same server, and `roger`, on `yow`, has mounted `/usr/tools`.

The `-a` option displays all the remote mounts in the form *hostname:directory*:

```
venus% showmount -a slugfest
yazoo:/usr/games
yow:/usr/tools
zippy:/usr/demos
venus%
```

8

Sharing Files Under RFS

As mentioned in Chapter 1, the SunOS operating system normally runs NFS as its system for allowing users to share files over a network. However, some sites running the SunOS operating system may elect to use RFS instead (or even in conjunction). Sharing files under RFS is similar to what you learned in Chapter 7, although the commands are somewhat different. Following is an introduction to mounting and accessing files under RFS. You may want to skim Chapter 7 to get a basic idea of what mounting is, what the file `/etc/fstab` is used for, and so on.[10]

filesystem is defined in Chapter 1.

Briefly, *mounting* a remote filesystem means making it look as though it were on your own machine. For example, suppose your company keeps a number of editing programs in a directory called `/usr/editors` on another machine called `zippy`. You (or your system administrator) can create your own directory `/usr/editors` on your machine, and then map the directory on `zippy` to your own `/usr/editors`. In this way, each time you refer to `/usr/editors`, you refer to the `/usr/editors` on `zippy`, as though it were on your own machine.[11] This mapping is known as mounting.

Under RFS, machines on a network are grouped into *domains*; a domain is a group of machines, ranging anywhere from one machine to all the machines on the network. You can mount filesystems from machines in both your own domain and domains to which you don't belong. Every domain has a *domain server*— a machine that keeps track of which filesystems are available for mounting, as well as password information for machines in the domain.

8.1. Advertising Resources With `adv`

In order for people to mount each other's filesystems, they must first *advertise* which filesystems, if any, are available for mounting. This is done with the `adv` command. The `adv` command puts the filesystem in the *domain advertise table* located on the domain server. An advertise table is a list of filesystems that have been advertised as available for mounting. The domain advertise table, then, is a list of all the available filesystems on all the machines in a domain.

---

[10] NFS was developed at Sun Microsystems. RFS was developed by AT&T.

[11] The `/usr/editors` on your machine is known as a *mount point*. It does not have to be the same as the directory on the remote system; you could, for example, mount `zippy`'s `/usr/editors` on `/home/venus/bin/editors`.

When you advertise a filesystem as available, you give it a special name that other people can identify it with. This special name identifies the filesystem as a *resource*. For example, if you're making the directory `/usr/graphics/tools` available, you can refer to it as the resource GRAPHTOOLS. Now anyone who wants to mount that directory can do so by mounting GRAPHTOOLS.

This is the format of the `adv` command (optional arguments are in brackets):

```
adv [-r] [-d "description"] resource pathname [clients]
```

These are the various options and arguments:

-r This means advertise this filesystem as read-only.

-d *description*
This is an optional description, up to 32 characters long and enclosed in quotes, of the resource being made available.

resource
This is the name you're assigning to this filesystem. It may be up to 14 printable characters long.

pathname
This is the full pathname to the filesystem you want to share.

clients
If you want to restrict the availability of the resource to only certain machines, you can list them here.

Here's an example of advertising a resource:

Note that you must be the superuser `root` to use most of these commands. See *SunOS User's Guide: Doing More.*

```
venus# adv -d "Computer Nerd Games" NERDGAMES /home/venus/games
venus#
```

In this example, the user on machine `venus` is advertising the filesystem `/home/venus/games` to be available as the resource NERDGAMES.

Displaying Your Advertised Resources

The `adv` command by itself gives a list of the resources on your machine that you've advertised as available for others to mount:

You do not have to be `root` to use `adv` without any arguments. (Note also that two of these lines are too long for the screen, and wrap around.)

```
venus% adv
NERDGAMES    /home/venus/games  read-write  "Computer Nerd Games"
 unrestricted
FILM         /home/venus/medici/personal read-only "video listings"
 zippy moondog
DOCTOOLS     /usr/doctools   read-only   "editors"  unrestricted
venus%
```

In the example above, the word "unrestricted" refers to the fact that anyone can mount DOCTOOLS and NERDGAMES, while only users on machines `zippy` and `moondog` can mount the other resource.

8.2. Mounting Remote Filesystems

Now that you've made your files available to others, how do you go about getting access to theirs? The first thing to do is to find out what's available — i.e., what resources other people have advertised as available for your use.

What's Available? The `nsquery` Command

The `adv` command, as illustrated above, showed you what was available for mounting from your machine; in other words, it displayed the *local* advertise table. The *domain* advertise table lists all the resources that everyone in your domain, including you, has advertised. Use the `nsquery` command to see this table.

`nsquery` by itself displays the entire domain advertise table. Here's how `nsquery` displays its results:

You do not have to be `root` to use `nsquery`.

```
venus% nsquery
RESOURCE     ACCESS       SERVER          DESCRIPTION

BUDGET       read-only    admin.manag     Old budget guesses
CHEWY        read/write   foods.krakow    Files on chewy foods
CLIENTS      read/write   admin.ducky     files on customers
DOCTOOLS     read-only    admin.venus     editors
EDITORS      read-only    admin.zippy     text tools
NERDGAMES    read/write   admin.venus     Computer Nerd Games
FILM         read-only    admin.venus     video listings
ROCKS        read/write   foods.krakow    Files on crunchy foods
venus%
```

You can also give `nsquery` an optional domain or machine name. There are three possibilities for doing so:

The command

`nsquery` *machine*

gives all the resources available on the named machine in your own domain. So typing `nsquery zippy` would show everything available on the machine `zippy`.

The command

`nsquery` *domain.*

(note the period) displays all the resources available in a *different* domain, so typing `nsquery graphics.` gives all the resources in the domain `graphics`.

Lastly, you can combine the two syntaxes: if you type

`nsquery` *domain.machine*

you'll see all the resources advertised by that machine in that other domain.

Here's an example of using `nsquery` to display all the resources on the machine `krakow` in the domain `foods`:

```
venus% nsquery foods.krakow
RESOURCE   ACCESS       SERVER          DESCRIPTION

CHEWY      read/write   foods.krakow    Files on chewy foods
ROCKS      read/write   foods.krakow    Files on crunchy foods
venus%
```

Unadvertising With `unadv`

You can remove a resource you've advertised from advertise tables with the `unadv` command. Just type `unadv` followed by the resource you want to remove:

```
venus# unadv NERDGAMES
venus#
```

The `mount` Command

Now that you know how to find out what resources are available, you can mount them.

The format of the `mount` command is (as always, optional arguments in brackets):

`mount [-r] -d` *`resource directory`*

where

resource
: is the resource on the other machine you want to access

directory
: is the directory on your machine you want to map the resource to (the *mount point*), and

`-r` indicates that you want the resource mounted read-only

```
venus# mount -d EDITORS /home/venus/medici/editors
venus#
```

Now anytime you refer to `/home/venus/medici/editors`, you'll have access to the editors which are on the machine `zippy`:

```
venus% ls /home/venus/medici/editors
SlowHand          WordGush          ex
Spel.tool         ed                vi
WordAgony         emacs
venus%
```

Displaying the Current Mounts

The `mount` command by itself will display all the filesystems you have mounted:

```
venus# mount
LOCAL_SUN4 on /mnt/local type rfs (ro)
venus#
```

Likewise, the `rmntstat` command tells you what resources of yours other people have mounted:

```
venus# rmntstat
RESOURCE  PATHNAME                    HOSTNAME

NERDGAMES /home/venus/games           buddha
TYPOS     /home/venus/medici/spell1   ducky garden
venus#
```

This example tells you that someone on the machine `buddha` has mounted your games directory, while machines `ducky` and `garden` have the `TYPOS` resource directory mounted.

Unmounting With `umount`

To unmount a filesystem, use the `umount` command. Just type

```
umount -d resource
```

where *resource* is the name of the resource you want to unmount.

8.3. The `/etc/fstab` File

You can have you machine automatically mount selected remote filesystems by putting them in your `/etc/fstab` file.[12] When you start your machine up, it will mount all the filesystems you list in `/etc/fstab`.

The format of the `fstab` file is given in Section 7.2.

[12] *fstab* stands for *file system table.*

Index

Notes

Systems for Open Computing™

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 37-29639

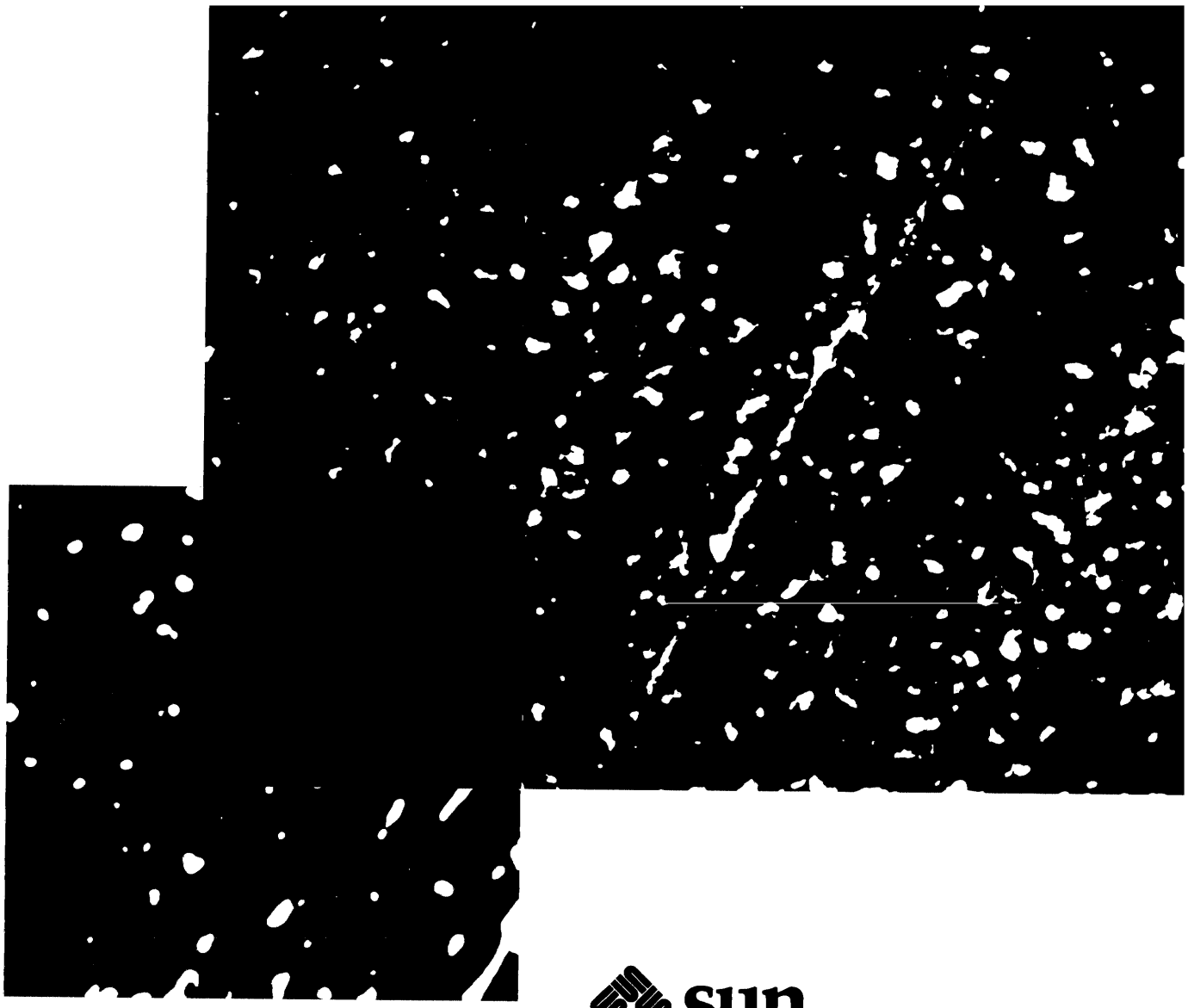
For U.S. Sales Office locations call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor, Green Lane
Bagshot, Surrey GU19 5NL
England
0276 51440
TLX 859017

Australia: (02) 413 2666
Canada: 416 477-6745
France: (1) 40 94 80 00
Germany: (089) 95094-0
Hong Kong: 852 5-8651688
Italy: (39) 6056337
Japan: (03) 221-7021
Korea: 2-7802255
New Zealand: (04) 499 2344
Nordic Countries: +46 (0)8 7647810
PRC: 1-8315568
Singapore: 224 3388
Spain: (1) 2532003
Switzerland: (1) 8289555
The Netherlands: 033 501234
Taiwan: 2-7213257
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 51440

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales



sun microsystems

Part Number: 800-3835-10
Revision A, of March 27, 1990