UNIX AT SLAC: GETTING STARTED[1]

Edited by M. Barnett, C. Boeheim, R. Cook, R. Dart, L. Moss, I. Vinson, L. White

January 29, 1997

Revision 2

PostScript file: /usr/local/doc/intro/unix-guide
Frame source file: /usr/local/doc/src/unix-guide/rev2

---

1. Adapted with permission from *UNIX at the SSC: Getting Started*, produced by the Superconducting Super Collider Laboratory (SSCL), 2550 Beckleymeade Avenue Dallas, Texas 75237 and edited by Jackie Damrau. The SSCL is operated by the Universities Research Association Inc. for the U.S. Department of Energy under Contract No. DE-AC02-89ER40486. This document was indirectly adapted from the document *Getting Started in UNIX*, produced and copyrighted by by the Academic Information Resources Group, Stanford University .

Contents

Tables

Figures

CHAPTER 1

Before You Begin

What is UNIX?

UNIX[1] is a multi-user, multi-tasking operating system originally from AT&T that runs on a wide variety of computer systems from micros to mainframes. UNIX is made up of the kernel, the heart of the operating system; the file system, a hierarchical directory method for organizing files on the disk; and the shell, the user interface through which you command the system.

The table below shows typical commands with their VM, Microsoft DOS, and VMS counterparts.

TABLE 1. Typical UNIX Commands and VM, DOS, VMS Counterparts

Command	UNIX	VM	DOS	VMS
List directory	ls	listfile	dir	directory
Copy a file	cp	copyfile	copy	copy
Delete a file	rm	erase	del	delete
Rename a file	mv	rename	rename	rename
Display a file	cat	type	type	type
Print a file	lpr	print	print	print
Check disk space	df	query disk	chkdsk	show device/full

Various UNIX systems also have graphical user interfaces (GUI) that provide a graphical, point-and-click interface to many of the above operations. Many vendors are now standardizing on the Common Desktop Environment (CDE) for that user interface.

1. UNIX is a registered trademark of X/Open Company, Ltd.

Why UNIX?

With its programming tools, application programs, and networking facilities, UNIX provides a powerful, versatile computing environment. Use of UNIX is widespread, and variations of the UNIX operating system are available on most types of hardware today, including mainframes, workstations, PCs, and embedded processors.

- UNIX is a general purpose, multi-tasking (able to handle multiple requests simultaneously), multi-user, interactive operating system.
- Since most of UNIX is written in a high level language, it has been relatively easy to install on many different computers, e.g., DEC, IBM, NeXT, and Sun.
- Licenses to install UNIX have been readily available, particularly to educational institutions. Some versions of UNIX are free.
- A wide variety of applications have been written to run under UNIX.

About This Document

UNIX at SLAC: Getting Started explains necessary UNIX concepts to help you learn to use basic UNIX commands, to run a few useful programs, and to manipulate files effectively. SLAC operates UNIX workstations on the IBM and Sun platforms. Information in this document applies to all of these computers. Graphical user interfaces may make it unnecessary to know UNIX commands for most tasks; you should refer to your system documentation if you are using a vendor GUI.

The first section of *UNIX at SLAC: Getting Started* takes you through an initial UNIX session. You will learn how to log in, enter simple commands, and log out. Subsequent sections describe the UNIX procedures most often used:

- Section 3, Working with Files (the unit by which the computer stores your work)
- Section 4, The Shell Program (the program that reads your commands and invokes programs to carry them out)
- Section 5, Editors
- Section 6, Printing

Throughout this document, sample sessions show what you will see during work sessions on the screen of your terminal, microcomputer, or workstation. The sample screen includes prompts and commands. The UNIX system uses a prompt to tell you it is ready to accept your next command. Often this prompt is your userid and the name of the computer followed by $; the prompt shown in the sample sessions here is just $.

Commands that you type are shown in a boldface monospaced font; what UNIX displays on the screen is shown in a regular monospaced font. Thus, in the following sample session, you would type **`date`** only. Text on the right side of the page is additional comments or more detailed instructions to help clarify the example. In this sample session, the text explains that you must press the RETURN key after typing **`date`**.

If what appears on the screen during a sample session is lengthy, it may be abbreviated within angle brackets to conserve space, e.g., <*system messages*>.

```
$ date                          After you type date, press the Return key.
Mon Nov 18 13:45:38 PST 1996
$
```

Questions

The *UNIX at SLAC* Web page at URL *http://www.slac.stanford.edu/comp/unix/unix.html* has additional information about the UNIX system at SLAC, sources of assistance, and links to further information. If you have questions about UNIX not answered in this guide or in the *UNIX at SLAC* Web page, contact the SCS Help Desk, 415-926-HELP (4357).

CHAPTER 2

Beginning UNIX

This section shows you how to login to SLAC's central UNIX system, begin and end a work session, inform the computer of the type of terminal you are using, change your password, and get help when you need it.

SLAC currently supports UNIX on two major architectures: SUNs and RS/6000s. In addition to SLAC's central UNIX system, some SLAC users use the Reason cluster of NeXTs, whose file system and available applications are somewhat different from the central UNIX system. If you are a NeXT user of the Reason cluster, you might want to read the document *Using Your NeXT at SLAC* by Paul Kunz. It is available online as file `/LocalLibrary/Documentation/Welcome.frame` if you are connected to the Reason Cluster, or as file `/nfs/ebnextk/LocalLibrary/Documentation/Welcome.frame` if you are connected to the central UNIX system.

UNIX Overview

Your UNIX Account

Normally people share a UNIX computer system. Each user must have a UNIX account, which provides a necessary means of regulating use of the system. An account includes a username and password. The username is the way you identify yourself to the system; the password verifies your identity. Accounts must be opened by filling out a *SLAC Computer Account Form* available at the Help Desk in the Computer Building Lobby. (Since the form is also available online as file `/usr/local/doc/forms/acctform.ps`, someone with access to the file can print you a copy).

Your UNIX account provides access to all SLAC UNIX machines; that is, the account allows you to log in and access your files from other UNIX machines on our network.

Making the Connection

First you must do whatever is necessary to get a "login prompt" from a UNIX host. You could, for example, walk up to a workstation that has "login:" as the last line on its display. It's not always that easy. Each of the following affect how you connect to UNIX and how you use it:

- Physical connection: ethernet, phone line (hardwired or dialup), ISDN;
- The type of machine attached to the keyboard you're using: UNIX workstation, X Terminal, PC compatible, Macintosh, ASCII terminal;

- Your machine's features: operating system (workstation/PC/Mac), terminal-emulator program (PC/Mac), terminal settings and features.

There are too many ways to connect to our system to describe them all here. The best source of advice, if you need it, may be someone in your group who uses similar equipment. You may also find what you need on the Web page at URL http://www.slac.stanford.edu/comp/net/residential.html. If these resources don't answer your questions,try the SCS Help Desk at 415-926-4357.

Clearly not all users will have the same system capabilities. However, most of the important UNIX functions can be performed independent of connection and equipment using standard command conventions.

Prompts

When the computer is ready to listen to your directions, it displays a prompt. This prompt varies depending on what system you are using. For SLAC, the *default* prompt is $. (*Default*, in computer terminology, is the action that the computer takes when you do not direct it to do otherwise.)

Commands

Commands are directions given to the computer by the user. Many command names are abbreviations for English words. To give the computer a command, you type the command name and press RETURN. An example of this was shown in "About This Document" on page 2

Arguments and Options

Most UNIX commands allow you to attach one or more arguments. Arguments determine the way in which the command carries out its function. Some types of arguments are names of computer files or time limits. For example, the command **`ls`** tells the computer to list the names of files in your current working directory. To find out whether the file name **`mbox`** is in your current working directory, you would type the argument **mbox** after the command **`ls`**:

```
ls mbox
```

You would then press RETURN to enter the command **ls** and its argument **mbox** into the computer. If the file **mbox** exists, its name would be listed on the screen.

An *option* is an argument included on the same line as the command and preceded by a hyphen (-). It tells the computer to execute a particular variation of the command. For example, **`ls -l`** (that is, the letter l) tells the computer to list the file names with additional information about each file. Options are sometimes called *flags*.

Special Keys and Control Sequences

There are several special key and control sequences that are helpful when you use UNIX. When you type a control sequence, you hold down the CTRL key while you type a certain letter. To type ^c, for example, you would hold down the CTRL key while you type c.

Consult the following chart when:

- you type a command and notice a typing error before you press RETURN or

- you want to stop a program that is running, or make other changes to a program.

TABLE 2. Special Keys and Control Sequences

You Want To:	Do This:
Erase the last character you typed	DELETE or BACKSPACE
Erase the last line you typed	^u
Erase the last word you typed	^w
Stop or interrupt a program currently running	^c
Signal the end of a file to a program waiting for input (e.g., in sending mail)	^d
Freeze the program's output scrolling on the screen, although the program may continue to run	^s
Unfreeze the output suspended by ^s	^q

WARNING!! UNIX is case sensitive. *Case sensitive* ***means that UNIX differentiates between uppercase and lowercase letters. For example, Open is not the same as open or OPEN. Pay special attention to case when you use UNIX.***

Logging In

Using a Terminal, Mac, PC, or X Terminal

You first must establish a connection through the Micom switch, a dial-in terminal server, or a software interface to one of the local UNIX hosts via an Ethernet TCP/IP line.

The following illustrates the procedure for establishing a dial-in connection to vesta03, one of our UNIX interactive hosts. Once the `login` prompt has been obtained, login can continue as shown in "Using a Workstation" on page 8:

```
    Unauthorized access is strictly prohibited!
                        SLAC Terminal Server

To connect to a host, type:
   telnet hostname    (or just hostname)
   switch        to connect to the Micom switch (until September 1997)

Logoff and drop the phone connection:
   logout

Escape to Terminal Server's xremote1> prompt:
   ctrl-^,x      (ctrl-shift-6 followed by 'x')
```

Notes:

```
(1) This service is restricted to connecting to SLAC hosts only.  To reach other
machines, first login to a SLAC host and telnet from there.
(2) The Micom switch service is scheduled to end September 1997.
```

```
(3) Xremote support is frozen.

xremote1> vesta
Translating "VESTA03"...domain server (134.79.16.9) [OK]
Trying VESTA03.SLAC.STANFORD.EDU (134.79.17.13)... Open

telnet (vesta03)

WARNING:  Unauthorized access to this computer system is prohibited.
          Violators are subject to criminal and civil penalties.
          All activities may be monitored and recorded and these records
          provided to law enforcement officials; by using this
          system you expressly consent to such monitoring.

login:
```

Using a Workstation

If you are using a UNIX workstation, the preceding connection sequence is not needed. Whether using a terminal or a workstation, logging in identifies you to the computer. Before you begin a work session, use your UNIX account to log in to the computer or workstation to be used.

Passwords are not displayed on the screen. This is just added protection provided by the UNIX system so that curious eyes behind you cannot read your password. By providing your username and password, you have completed the log in.

If you make a mistake, type your login name and password again. Once you successfully log in, the system displays its prompt and waits for your commands

```
login: meb                                  Enter userid and press CR

Enter password for meb:                     Enter password and press CR

User meb kerberos-authenticated via AFS 3.2.

IBM AIX Version 3.2                         These messages will vary.


WARNING: Unauthorized access to this computer system is prohibited.-
Violators are subject to criminal and civil penalties.


Last unsuccessful login: Mon Aug 16 12:40:04 PDT 1993 on pts/55

Last login: Wed Sep  1 12:16:10 PDT 1993 on pts/26

meb@ganymede $                              Normal command prompt
```

Workstations and Window Systems

If you use a workstation, you will probably want to start a *window system* on your machine as soon as you log in. A window system is a software package that makes it easy for you to take full advantage of the graphics and windowing capabilities of a workstation (such as multiple active sessions, high-resolution fonts and graphics).

Several window systems are available for each type of workstation. The most popular is X Windows, some variant of which is available for every workstation at SLAC. There are others, however, and your particular needs may require a different window system. Again, the best advice may come from someone in your group.

Setting Your Terminal Type

Your terminal type tells the computer how to format output for your display and how to interpret keyboard entries. This is especially necessary for the keyboard and display interaction used by text editing programs. Even if you are using a workstation, it is a good idea to tell the computer what kind of terminal emulation to use.

Each time you begin a new computer session, you should set the terminal type. To make setting the terminal occur automatically at the beginning of every work session, follow procedures detailed in "Customizing the Session: Defining the Shell Environment" on page 30.

At SLAC, chances are that *vt100* is the terminal type to specify. There are exceptions, so if you are not sure what kind of terminal you are using, ask your computer support coordinator or the Help Desk at 415-926-4357. To set your terminal type, use the command **`set`** with the arguments **term=***terminal type*.

`$ set term=vt100`	Type **set term**=*terminal type* RETURN.
`$`	In this example, the terminal is a vt100.

Changing Your Password

Please change the password that is assigned to you when you open your UNIX account. Although you will not be required to change your password periodically, as users are on the VM system, it is a good idea to change it, perhaps, every six months to preserve the security of your account. The procedure for changing the password varies depending on the machine architecture, but it is consistent for our supported UNIX machines.

In changing your password, the new password should *not* be any word found in the English dictionary, your username, or any permutation of your name or initials. Helpful do's and don'ts and ways of constructing passwords are given in the file `/usr/local/doc/policies/password`. Printing of files is described section "Printing" on page 43.

Changing Your Password on Any SLAC UNIX System

To change your password, login to any UNIX host. Then use the command **`password`**.

`$ login unixhub`	Login to the UNIX host unixhub, RETURN.
`$ password`	Command to change your password, RETURN.
`Changing password for ilse.`	
`Old password:`	Type your old password, RETURN.
`New password:`	Type your new password, RETURN.
`Retype new password:`	Type your new password again, RETURN.

The **`password`** command changes your UNIX login password as well as your AFS password. If you do not have an AFS account, the **`password`** command creates one for you and sets the password to your UNIX login password. Re-

member that, while changes to your AFS password are immediate, changes to your UNIX login password do not go into effect for up to one hour.

Logging Out

Logging out ends your work session on the computer. Finish each work session by logging out to ensure that no one else can continue working on the computer using your account. If a message on the screen indicates there are stopped jobs, be sure to kill them before logging out. (See "The Shell Program" on page 29.)

```
$ logout
```
Type logout, press RETURN.

Getting Information

The UNIX Manual

On traditional UNIX systems, the UNIX reference manual is online (that is, in the computer) and is commonly referred to as *man pages*. The manual contains descriptions of the UNIX commands, so you can refer to them on your screen as needed. Unfortunately, you may have to look around to find just what you are looking for. Since the online manual is organized by command, as a novice you may find it hard to know which command you need to read about. The **man** and **apropos** commands can help you. As shown in Figure 1 below, you can use the **-k** option for **man** to obtain a list of topics related to a particular key word.

Finding Shell Documentation

Most UNIX commands run programs; you can find documentation for almost all of these commands with the **man** command. However, some commands are built into the shell program; you can find documentation for these commands with **man tcsh** (or whatever shell you use). For more information about the shell program, see "The Shell Program" on page 29.

Directory /usr/local/doc

This directory contains documentation, produced at SLAC and elsewhere. You might want to browse through this directory when you are looking for information on a particular topic.

UNIX Journal Club

The UNIX Journal Club initially met in September 1990. When announced, the stated purpose of the Club was “to share UNIX experience at SLAC and to be a self-teaching society on UNIX issues...The club is not intended to be a UNIX committee meeting nor the equivalent of the CCG or VCG.”

The Club originally met once a month but no longer meets regularly. Announcements of upcoming meetings are physically posted around SLAC and electronically in the Netnews group slac.users.unix (see "NetNews" on page 54) and in VM News announcements. Handouts and lecture notes of past UNIX Journal Club meetings are available at the Help Desk in the Computer Building Lobby.

World Wide Web (WWW)

The Web is a network-based information retrieval system initially developed at CERN. It makes a vast array of information from around the world available to you by using a single easy-to-use interface. Better still, you don’t need to

know how or where the information is actually stored to access it; the Web's software takes care of that. In fact you don't even need to know that the information exists at all. The Web makes it easy to browse available information and discover new sources of information.

To access the Web you use a program called a *browser*. Browsers are available for all UNIX platforms (as well as non-UNIX platforms). The Web presents information in the form of hypertext, which are small documents that contain links to related documents. It is particularly suitable for High Energy Physics, where rapid sharing of up-to-date information among widely spread collaborations is essential.

At SLAC most of the SPIRES databases, such as the HEP publications database, BINLIST, HEPNAMES (email addresses of physicists worldwide), SLAC seminars and many more are accessible. Information on specific experiments is also available, including SLD and BES. In addition, many departments, including Slac Computing Services, now provide much of their documentation, news, and other communications on Web pages. We suggest that you learn how to access information on the Web as soon as you can.

To access the Web on a UNIX workstation or X-terminal, enter the command **`netscape &`** after you have started X-windows. From an ASCII terminal or a terminal window, enter the command **`lynx`** to invoke a character-mode Web browser. Lynx will provide usable access to most of the important documents on SLAC's Web pages.

For questions and suggestions about the Web, contact the WWW Support Coordinator (WSC) for your group or Division. Names of WSCs are given at the Web URL http://www.slac.stanford.edu/slac/www/wsc/wsc-list.html.

Resources for Answering UNIX Questions

Resources at your disposal for answering UNIX questions include the Help Desk, the online AID facility, and the online **man** pages.

The locally written **aid** command lets you search for online information and/or pointers to information about UNIX or SLAC in general. To use the command, type **`aid`** *keyword ... keyword.*

```
$ aid unix

Matching 'unix'

-> Suggestions from /usr/local/lib/aid/unix:
See /usr/local/doc/FAQ/unix/intro for beginner's questions about UNIX.
Try the /usr/local/doc/intro directory.
BSD - Berkeley Software Distribution (UC Berkeley's version of UNIX).
> BSD UNIX is the basis for Sun's and NeXT's versions of UNIX.
> (Although Sun is moving toward System V UNIX.)
Print /usr/local/doc/forms/acctform.ps to a PS printer (computer account form).
See /usr/local/doc/usc-names for list of UNIX Support Coordinators.
See /usr/local/doc/usc-job for a description of USC duties.
See /usr/local/doc/xcg for information about the XCG.
'xCG' stands for 'experimental UNIX Group.'
'/usr/man' contains online version of the UNIX reference manual.
$
```

You can contact the Help Desk at 415-926-HELP (4357) for questions that neither the **aid** command nor the **man** pages can answer.

```
Entering "man -k <word>" produces lines from the various man pages that contain
<word> in their NAME-descriptor lines.  ("apropos <word>" produces the same
```

```
output.)

$ man -k password

password (1)            - change Unix/NIS and AFS passwords
passwd, chfn, chsh (1)  - change local or NIS password information
yppasswd (1)            - change your network password in the NIS database

[Several additional lines were produced.  These, with "(1)," are commands,
which are documented in section 1 of the UNIX manual.]

$ man password

PASSWORD(1)              USER COMMANDS               PASSWORD(1)

NAME
      password - change Unix/NIS and AFS passwords

SYNOPSIS
      password [ -f | -s ] [ username ]

DESCRIPTION

      password changes the Unix/NIS and AFS passwords. A prompt is issued for
      the old password and, if it is correct, two prompts are issued for the
      new password.  If the new password is 6- to 8-characters long and both
      responses match the Unix password in the Network Information Service
      (NIS) database is changed.  The local operating system may impose addi-
      tional constraints on the new Unix/NIS password format (e.g., minimum
      number of different characters).

       .
       .
       .
SEE ALSO

     afsacct, kpasswd, passwd, ypfiles, yppasswd, yppasswdd

       .
       .
       .

      [man pages contain several sections on what the command does, how to
      enter it with its various options, and even bugs.  A very useful section
      is "SEE ALSO"; if you can't think of a command's name, often the man page
      of a related command will give you the reminder.]
```

Figure 1. Sample man Output

CHAPTER 3

Working With Files

Structure

Computer files can be thought of as folders inside a file cabinet. Each file is labeled with a name. Files are quite important allowing information storage for use at a later time. After you have finished your computer session, the computer forgets what you have been doing, unless your work is saved in a file.

UNIX files are organized in directories, similar to the file folders discussed above. Directories are files, but instead of containing data or text as other files do, the directory contains information about other files. Directories are organized like inverted trees. The root directory named / is at the top. Branching out from the root directory are the rest of the directories. Any directory can contain both directories and other files.

All directories together are referred to as a *file system*. The SLAC UNIX file system is a single file system in one sense, that is, all directories originate directly and indirectly from a single root directory at the top. However, in another sense, we have two file systems: the Network File System (NFS) and the Andrew File System (AFS) that share that common root directory as shown in Figure 2. SCS started with NFS support and began moving toward AFS because it offers significant advantages over NFS.

The most notable difference between NFS and AFS is how file access permissions (that is, who is allowed to read, write, execute, etc. a given file) are implemented. The discussion "Handling File Security: File Access Permissions" on page 25 assumes NFS, not AFS. How file access permissions work in AFS is described in the document *SLAC AFS Users' Guid*e, available at the Web URL *http://www.slac.stanford.edu/comp/unix/afs/users-guide.html* or at the SCS Help Desk in paper form. Be aware that even if your home directory is in NFS, you may still need to become familiar with AFS because most collaboration data space is in AFS and SLAC's production WWW space is also in AFS.

.

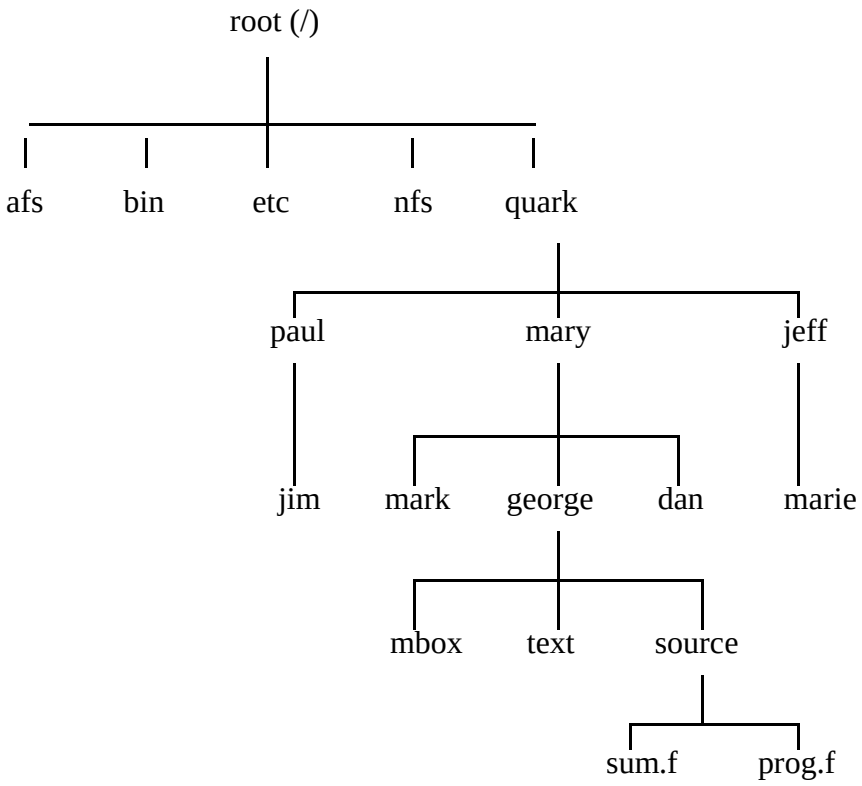

Figure 2. Sample Directory Structure

The Root Directory

The / (slash) at the top of Figure 2 is the root directory. Within this root directory the following files are shown: `afs, bin, etc, nfs, quark`. These files just happen to be directories, which contain their own files. These directories-within-a-directory are called *subdirectories*. Thus, `afs` and `nfs` are subdirectories of /. Every file on the system is contained in the root directory or in one of its subdirectories.

Absolute Pathnames

To identify a certain file in the file system, you can specify an absolute pathname by telling UNIX where the file is in relation to the root directory. For example, to refer to the file `sum.f` above, use

```
/quark/mary/george/source/sum.f
```

The first / signifies the root directory. Each subsequent directory name is separated by an additional /.

Your Current Working Directory

Whenever you are using a UNIX system, you are in a specific directory. That directory is called the current working directory. For example, if you were in the `/afs` directory, your working directory would be `/afs`. If you were in the `/quark/jeff` directory, your working directory would be `/quark/jeff.` Commands that operate on directories use the working directory, unless specified otherwise.

Relative Pathnames

Besides identifying a file by its absolute pathname, you can use a relative pathname. That is, you can tell UNIX where the file is in relation to the working directory. Suppose the working directory is `/quark`, as shown in the example above. To refer to the file `sum.f`, you could specify its relative pathname `mary/george/source/sum.f`. This tells the computer to look in the working directory for a subdirectory named `mary`. Having found the `mary` subdirectory, the computer looks for the `george` subdirectory. In the `source` subdirectory, the computer should look for the file `sum.f`. Notice that there is no / preceding `mary`. If a / preceded `mary`, the computer would have looked in the root directory for a subdirectory named `mary`. **Relative pathnames never begin with / because their origin is the working directory, not the root directory**.

Your Home Directory

When you first login to a UNIX system, you are in the directory that holds your personal files, known as your *home directory*. Your home directory takes the form /**u**/*gg*/*username* or more simply **~***username*, where *gg* is your group code and *username* is your UNIX userid. Because SCS is currently supporting both NFS and AFS file systems, some users have their home directory in AFS; others still have their home directories in NFS. If you are not sure whether your home directory is in NFS or AFS, enter the command: **`ypmatch`** *username* **`homes`** Substitute your UNIX username for *username*. For example, user ilse's home directory is in AFS:

```
ilse@odysseus $ypmatch ilse homes
ilse:/afs/slac.stanford.edu/u/sf/ilse
```

but user billie's home directory is in NFS:

```
ilse@odysseus $ ypmatch billie homes
billie:/nfs/juno/u8/sf/billie
```

Periods as Shorthand

When you want to use the working directory as an argument for a command you can use the period (**.**) as a shorthand notation. Suppose the present working directory is `/quark/paul`; you could refer to it with **.** as an argument for the copy command **cp**:

```
cp /quark/mary/george/source/sum.f .
```

In this case, the file `/quark/mary/george/source/sum.f` would be copied into the working directory `/quark/paul`. See "Copying Files" on page 21 for more information about the **cp** command.

The shorthand term for the directory one level above the working directory is two consecutive periods (**..**). For example, if your present working directory is `/quark/paul`, **..** would refer to the directory `/quark`. You could use **..** as an argument for the change directory command:

```
cd ..
```

Pressing RETURN

The previous sample sessions reminded you to press RETURN after typing a command and its arguments, but the sample sessions in the rest of this document assume that you know to press RETURN to enter a command.

Symbolic Links and the `sl` *(Show Symbolic Links) Command*

You will undoubtedly come across instances of symbolic links, which are basically pointers in one location of the file system to the actual location of a file in another part of the file system. For example, in my home directory (`/afs/slac/u/sf/ilse`) I have the symbolic link `afs-slac` that points to `/afs/slac.stanford.edu`. From there I can quickly get to the SLAC WorldWideWeb (WWW) production space when I want to work on WWW files.

Given a symbolic link, if you want to know the absolute pathname of the file, use the **`sl`** (show chain of symbolic links) command.

Example 1:

```
ilse@odysseus $ sl afs-slac

afs-slac:
afs-slac -> /afs/slac
/afs/slac -> slac.stanford.edu
     slac.stanford.edu
```

Symbolic links to absolute pathnames (absolute pathnames start with a slash (/); relative pathnames don't) start over at the left margin. Symbolic links to relative pathnames are aligned vertically with the path element they replace. So the easiest way to read this sl output is to substitute `slac.stanford.edu` in the last line for the path element `slac` directly above it in the last and only absolute path `/afs/slac` yielding an absolute pathname of `/afs/slac.stanford.edu` .

Example 2:

```
ilse@odysseus $ sl /usr/local/bin/environ

/usr/local/bin/environ:
/usr/local/bin -> bin.next
           bin.next -> /a/juno/usr/local/bin.next
/a/juno/usr/local/bin.next -> bin.next-m68k
                  bin.next-m68k/environ -> ../bin.shared/environ.main
                  bin.shared/environ.main
```

To decode the absolute pathname in this example, substitute the last line,`bin.shared/environ.main`, for the path element `bin.next` in the last absolute path `/a/juno/usr/local/bin.next`, so that it reads: `/a/juno/usr/local/bin.shared/environ.main`

Handling Directories

This section covers identifying the working directory; finding out what files and subdirectories are contained in a directory; changing directories; creating directories; and removing directories.

Identifying the Working Directory

Again, the directory you are presently using is called the (current) working directory. Most commands that operate on directories use the working directory, unless you specify otherwise. Therefore, it is important to know what the working directory is; to find out, use the print working directory (**pwd**) command.

cd /nfs/juno/work	Changes the current working directory to /nfs/juno/work
pwd	Prints the working directory, that is, lists the name of the working directory on the screen.
/a/juno/work	The pwd command almost never shows the pathname exactly as you would use it on the **cd** command. In this case /nfs/juno/work is a symbolic link to /a/juno/work (see section "Symbolic Links and the `sl` (Show Symbolic Links) Command" on page 18).

Note: *When this guide shows you a pathname that is different from the version shown by the* **pwd** *command, please use the name shown in this guide.*

WARNING!! Never use a pathname that starts with "/a/" because "/a/" is temporary and may disappear in five minutes. Use /nfs/ in place of /a/ at the start of the absolute pathname.

Finding Out What Is in a Directory

List Directory Command

The **ls** command is used for listing the files and subdirectories contained in a directory. As shown in Figure 3, different information displays depending on what arguments you use.

Note: *The commands shown in this chapter have additional options. For more information, give the* **man** *command (see section "Resources for Answering UNIX Questions" on page 11).*

Dot Files

Within your home directory are some files whose names begin with a dot (or period). Most users do not need to access dot files often. They do not show up on the list supplied by the computer when you type the **ls** command (assuming you do not use the option **-a**). Because they are not listed on the screen of your terminal, you can find other files more easily. Common dot files are listed in Table 3 below**.**

TABLE 3. Common Dot Files

`.cshrc`	The system runs the commands in this file each time you start the C shell. This varies, though, depending on the shell you use. Usually a good place for alias commands. (See section 4.2.)
`.forward`	Contains address(es) where mail for this account should be forwarded (e.g., ilse@slacvx.slac.stanford.edu or simply ilse@slacvx).
`.login`	The system runs these commands when you login. (See section 4.1.) This varies, though, depending on the shell you use.
`.logout`	The system runs these commands when you logout.This varies, though, depending on the shell you use.
`.plan`	Information you leave in this file displays when your username is fingered. (See section 8.2.)
`.project`	Information you leave in this file displays when your username is fingered. (See section 8.2.)

ls (which is a SLAC alias for **ls -F**) lists the names of files or subdirectories in the working directory. UNIX alphabetically orders files and directories with initial uppercase letters first, then with initial lowercase letters.

```
Library/      Mailboxes/      memos/
$
```

ls -a lists all files, including dot files. The character / after a name indicates a directory; * after a name indicates it is an executable program**.**

```
./              .cshrc          .newsrc          Mailboxes/
../             .kshrc          .profile         memos/
.NeXT/          .login          Library/
```

ls /usr lists the names of files or subdirectories in a directory other than the current one, here, /**usr**.

```
adm/     etc/            local/       prman/      standalone/
apps/    filesystems/    man/         pub/        template/
bin/     games@          netware/     share@      tmp/
cad/     include/        nfs/         shlib/      ucb/
dict/    lib/            preserve/    spool/
```

For more information about each file, use the **-l** option. **ls -l** lists each file or directory in the working directory, along with its permissions (security information), creator, size in bytes, and date of last modification.

```
$ ls -l
total 12

drwxr-xr-x  9 ilse        2048 Aug 10  1994 Library/
drwxr-xr-x  2 ilse        2048 Apr 17  1995 Mail/
drwxr-xr-x 42 ilse        4096 Oct 20  1995 Mailboxes/
-rw-r--r--  1 ilse         885 Apr 27  1994 datasave.opt
```

Figure 3. ls Command

Changing Directory

`$ pwd` `/a/juno/u8/sf/cathie`	Prints working directory (i.e., lists the name of the working directory on the screen).
`$ cd ..`	Changes to the directory one level above. A double period is a special name for the directory one level above.

Creating Directories

For organization purposes, you may want to create a new subdirectory to hold a group of related files. The **mkdir** command is used to make a new directory. Once the new directory is created, you can either create files in that directory or move existing files into it. For details on moving files into a directory, see section "Copying Files" on page 21.

`$ mkdir test`	Directory name is `test`. You can specify the directory by its absolute or relative pathname.

Removing Directories

If you need to remove a directory, make sure it is empty. You can use **`rmdir`** to remove a directory only if the directory contains no files. To remove ordinary files, see "Removing Files Permanently" on page 22.

`$ rmdir test`	In this example, the directory to remove is `test.` You can specify the directory by its absolute or relative pathname.

Handling Files

This section covers copying files; moving files or changing their names; creating files; and removing files permanently.

Copying Files

The **cp** command copies files. The first argument is the source file (i.e., the file you are copying). The second argument is either the name you want for the new copied file *or* the destination directory. If a directory is given as the second argument, the file is copied into that directory with the same filename. (For a discussion of arguments, see "Beginning UNIX" on page 5.) Groups of files can be copied by using special characters (see "Referring to Groups of Files" on page 36).

After the source file is copied, it remains intact in the directory where it was located.

`$ cp sum.f sample.bak`	Copy the file `sum.f`. The name of the new copied file is `sample.bak`.
`$ cp /quark/paul/.cshrc .`	Copy the `.cshrc` file to the working directory. The second period indicates the working directory.

WARNING!! When you use the `cp` command without the `-i` option, the computer does not tell you whether a file already has the name you chose for the destination file. If that file name already exists, the computer removes the existing file before it copies the file you specified.

Moving and Renaming Files

The **mv** command is used to move a file from one directory to another or to rename a file. The first argument is the source file (i.e., the file you are moving or renaming). The second argument can be the destination directory (the directory where the file should be moved) *or* the new name of the file (use the absolute or relative pathname). If a directory is given as the second argument, the file is moved into that directory with the same filename. (For a discussion of arguments, see "Beginning UNIX" on page 5.)

Unlike **cp**, if you move a file or change its name, the source file is deleted from its directory after the procedure is finished.

```
$ mv sum.f test.case
```

Gives the file sum.f the new name test.case (i.e., renames the file).

```
$ mv test.case /usr/smith/cases/test.bak
```

Moves the file `test.case` to the file `test.bak` in the subdirectory `cases`.

Creating Files

When you use a program, such as a text editor, you create a file to save your work. That way, the file can be opened later to allow for editing, printing, or adding to its contents. (See "Editors" on page 39 for information on creating a file during text editing.) There are other ways of creating files. Programs sometimes generate files, and a file may be created to store the output of a command.

Removing Files Permanently

WARNING!! You cannot recover a file that has been removed by this method!

The command **rm** is used to remove files for good. To protect against unintentional destruction of a file, you can use the option **-i** with the **rm** command. The computer will ask whether you are sure you want to remove this file before proceeding. Use the absolute or relative pathname of the file you want to remove.

`$ rm test.case`	Removes the file `test.case` permanently.
`$ rm -i test.case`	Remove the files `test.case`; requests confirmation.
`rm: remove test.case? y`	Type **y** for yes or **n** for no.
`$ rm -i test*`	A more instructive example of the use of the **-i** option is this case, where multiple files begin with 'test', some of which you want to remove and some of which you don't.
`rm: remove test.case? y`	
`rm: remove test1.case? n`	

Browsing Through Files

There are three common ways to browse through a file: looking at a file a screenful at a time, looking at the beginning or end of a file, and using an editor (see "Editors" on page 39.)

Looking at Files One Screenful at a Time

You can use the **more** command to display a screenful of a file at a time. To display the next screenful, press the space bar. The **more** command scrolls the text as it displays the next screenful.

```
$ more memo
```

Scrolls the contents of the file `memo`, screen by screen. Press the space bar to display the next screenful. To return to the prompt, type **q** or hold down the CTRL key and type **c**.

```
<contents of memo>
$
```

Looking at the Beginning or End of a File

Sometimes you want to see just a few lines at the top or bottom of a file to check on its contents. The **head** command displays the first ten lines of a particular file, and the **tail** command displays the last ten lines of a file. Both allow you to specify the exact number of lines to display with the **-n** option (where **n** is the number of lines to display) (see Figure 4).

Searching and Comparing Files

Searching for Text

Use the **grep** command to search for text strings or consecutive words in a file. You supply the string to search for and the file(s) to search in. If the string you are looking for contains spaces, you must put the string in single or double quotes. With the **grep** command, any lines in the file containing the string display on the screen.

```
$ grep Computing test.doc
```

Searches `test.doc` for text string *Computing*.

```
to carry out the SSCL mission. This plan introduces the Computing and The Computing and Communications
Organization within Computing and Communications to concentrate efforts in these areas
```

Comparing Files

The **diff** and **comm** commands allow you to compare two files to see whether they are identical. The **diff** command displays the actual lines in each file that differ. The **comm** command compares two alphabetically *sorted* files and produces a 3-column output: column1 shows lines only in file1; column 2 those lines only in file2; and column 3 lines in both files.

```
$ diff rose1 rose2
```

Displays differences between files.

```
< Violets are blue
```

Line in `rose1` differs from `rose2`.

```
-
> Violets are green
```

Line in `rose2` differs from `rose1`.

```
$ ilse@odysseus $ comm sg1 sg2
        abh
                bobcook
                esr
joann
kls
les
         ljm
                lmwhite
                meb
                randym
                renata
                wcw
                wglp09
```

Comparison results of files `sg1` and `sg2` are shown in three columns. The columns are offset by tabs. Column1 shows lines only in `sg1` ; column2 only those lines in `sg2`; column3 lines in both files.

`$ head /usr/local/doc/policies/password` Displays the first 10 lines of the file `/usr/local/doc/policies/password`

```
This is file /usr/local/doc/policies/password.

This article on choosing good passwords is by Lionel Cons of group CN/SW at CERN.
It appeared in CERN Computer Newsletter 210.  Mr. Cons kindly allowed us to adapt
it for SLAC users.

Introduction
------------

A good password:
```

`$ head -5 /usr/local/doc/policies/password` Displays the first 5 lines of the file `/usr/local/doc/policies/password`

```
This is file /usr/local/doc/policies/password.

This article on choosing good passwords is by Lionel Cons of group CN/SW at CERN.
It appeared in CERN Computer Newsletter 210.  Mr. Cons kindly allowed us to adapt
it for SLAC users.
```

```
$ tail /usr/local/doc/policies/password   Displays the last 10 lines of the file
                                          /usr/local/doc/policies/password

together with a punctuation character between them (or a digit if you can only
use alphanumeric characters). For example: 'dog+F18' or 'comP7UTer'. Note that
'dog', 'F18' or 'computer' are in dictionaries but as the passwords use
punctuation or digits, mixed-case character, they are really hard to guess.

Remark: if you use mixed-case characters, do not use the following methods:
 - all lowercase or all uppercase
 - only the first or the last character in uppercase
 - only vowels in uppercase
 - only consonants in uppercase.

$ tail -5 /usr/local/doc/policies/password  Displays the last 5 lines of the file
                                            /usr/local/doc/policies/password

Remark: if you use mixed-case characters, do not use the following methods:
 - all lowercase or all uppercase
 - only the first or the last character in uppercase
 - only vowels in uppercase
 - only consonants in uppercase.
```

Figure 4. Head and Tail Commands

Handling File Security: File Access Permissions

The following discussion applies to files in the NFS system. While file access permissions in the AFS file system work differently, AFS does not ignore the nine UNIX bits described below entirely. Therefore, even if you do not have to deal with files in the NFS file system, it is still a good idea to become familiar with the way file access permissions work under NFS.

UNIX Security

UNIX allows you to protect your work as you see fit. You decide what you can do to it, what the group you belong to can do to it, and what others can do to it. With the UNIX security system, you can determine who can see, alter, and use your files and directories.

About Security

In the UNIX environment, those who may access a particular file or directory are divided into three classes as shown in the following chart.

Class of User	Definition
user	The user who owns the file/directory.

Class of User	Definition
group	The group of users who share access of a file. (Every user belongs to one or more groups. To see what groups you are in, type **groups** at the prompt.) Each file and directory "belongs to" one user and one group.
others	All other users.

Each file or directory has a set of permissions that control access to it. For each class of users above, there are three permissions that may be granted to that class for each file or directory. These are shown in the following chart.

Permission	File	Directory
read (r)	Allows class to read the file.	Allows class to list the names of files and subdirectories in the directory.
write (w)	Allows class to modify a file (e.g., to append to it or to truncate it).	Allows class to create and remove files and subdirectories from the directory.
execute (x)	Allows class to execute a file.	Allows class to access files in the directory if they know their names.

When you create a file or directory, it is assigned a default set of permissions, which you can change. With **ls -lg**, you can see the permissions associated with a particular file or directory as the first 10 characters on each line. (Be sure to use the correct absolute or relative pathname for the file or directory.)

```
$ ls -lg
drwxr-xr-x  3  ilse    sf    1536 Sep 24 10:15 Telecom/
-rw-r--r--  1  ilse    sf    830 Aug 23 12:39 Unix-guide-questions
-rw-rw-r--  1  ilse    sf    353280 Sep 27 10:34 Unix-guide-
                             slac.block.frame
-rw-rw-rw-  1  ilse    sf    353280 Sep 29 11:00 Unix-guide-
                             slac.block.frame.auto
```

The **-g** option shows the group to which the file or directory belongs; in this example, the group is `sf`. Note, however, that **ls -lg** means show the group on a Sun, but **ls -l** means show the group on an RS/6000.

The first character indicates whether this file is a directory (**d**) or just a regular file (–). The next three groups, consisting of three characters each (nine characters total), are the permissions granted for the user, group, and others classes, respectively. A permission is granted if the letter is present and not granted if the (–) is there instead. Thus, in the example above, the file `Unix-guide-questions` is a regular file (–) that allows the user reading and writing privileges (**rw-**), but limits the group and others to only reading the file (**r- -**).

Making Security Changes

The **chmod** command allows you to change the permissions associated with a file or directory that you own.

```
$ ls -lg
total 12
-rw-r--r--  1 bobcook  sf    2847 Sep   6 11:42 groupfile
```

```
drwxr-xr-x  2 bobcook  sf      32 Sep   6 11:41 privatedir/
-rw-r--r--  1 bobcook  sf     664 Sep   6 11:42 prog
$ chmod a+x prog
$ chmod g+w groupfile
$ chmod go-xr privatedir
$ ls -lg
total 12
-rw-rw-r--  1 bobcook  sf    2847 Sep   6 11:42 groupfile
drwx------  2 bobcook  sf      32 Sep   6 11:41 privatedir/
-rwxr-xr-x  1 bobcook  sf     664 Sep   6 11:42 prog*
$
```

Backing Up and Retrieving Files

Backup and retrieval of NFS files and AFS file are handled differently. See the WWW UNIX Backups page at *URL http://www.slac.stanford/comp/unix/backup.html* for details.

CHAPTER 4

The Shell Program

After you log in, the shell program, or shell, is activated. A shell is a user interface program through which you can communicate with UNIX. The shell displays the prompt $, which is changeable (see "Customizing the Session: Defining the Shell Environment" on page 30), as a signal that it is ready to receive your UNIX commands. It interprets and executes the UNIX command that you type, and when done, gives you the next prompt. The default shell for SLAC UNIX accounts is either the C shell, csh, for older accounts; or an extended version of the C shell, tcsh, for accounts created more recently. Both these shells use a syntax based on that of the C programming language (hence their names). The features of tcsh are a superset of those of csh. One of the most popular additional features in tcsh is an intuitive (and VMS-like) interface for recalling and editing previously-issued command (see the section "Keeping Command History" on page 37). You can obtain full documentation about the C shell with **`man csh;`** for the additional features of tcsh, use the command **`man tcsh`**. (For more information on the **`man`** command, see section "Resources for Answering UNIX Questions" on page 11.)

Other shells available at SLAC include the Bourne shell, the Korn shell (ksh) and the Bourne Again shell (bash). The latter two are each upwardly compatible extensions of the Bourne shell, which was the original UNIX shell; all three share a syntax that is somewhat different from that of csh and tcsh. On most systems, the Bourne shell is synonymous with the “default shell”, and is referred to simply as “sh”; however, IBM’s version of UNIX, AIX, uses the Korn shell as the default, and this is assumed in all the IBM documentation. Moreover, starting in Version 4 of AIX, the command name “sh” invokes the Korn rather than the Bourne shell. In order to invoke the Bourne shell on an AIX 4 system, you must use the “bsh” command (note that the bsh command does not exist in all versions of UNIX).

Although shells normally read one command at a time from your terminal, they can also read commands from a file. Such a file is called a ***shell script***, and may include a variety of control structures, such as loops and if-then-else blocks. Thus, in addition to a user interface, the shell also provides a simple programming language. For more information about the programming features of a particular shell, see the appropriate **`man`** page.

Unless otherwise noted, the syntax and features of tcsh will be assumed throughout the remainder of this chapter.

Changing Shell Programs

During a work session you may from time to time start a new shell. This may be done explicitly, for example to run a script written for a different shell language, or implicitly when you use certain commands (e.g., the **`script`** command, described in "Keeping Command History" on page 37). A change in the default prompt may indicate that you

have invoked a new shell. Your *login shell* is the shell that is automatically invoked when you first login to a UNIX system. To leave a shell, use the **exit** command or press CTRL-d. This will return you to the previous shell or perform a logout if you are in your login shell

Shell programs provide methods to initialize your environment when they detect that they are being invoked as login shells, but these methods differ. The only shells supported by SCS as login shells are tcsh and csh; the former is recommended, and the latter is supported primarily for backward compatibility. To change your login shell you must run the **chsh** command on host unixhub (the change may take an hour or longer to actually go into effect):

```
ilse@janus $ rlogin unixhub                                  Log in to UNIX host unixhub.
Password:                                                    Type your password.
Last login: Tue Aug 24 08:44:25 from JANUS.SLAC.Stanf
SunOS Release 4.1.2 (SERVER) #1: Tue Feb 23 18:52:48 PST 1993
WARNING: Unauthorized access to this computer system is prohibited.
Violators are subject to criminal and civil penalties.
ilse@unixhub $ chsh                                          Change shell command.
Changing NIS login shell for ilse on unixhub.
Old shell: /bin/csh
New shell: /bin/tcsh                                         Type full pathname of new shell.
Password:                                                    Type your password.
```

Assuming you're switching between the csh and tcsh shells, you must also modify the `.cshrc` file in your home directory once the above change has gone into effect (See the next section for more information about the `.cshrc` file). Open this file in a text editor (see "Editors" on page 39 for information about text editors) and look for the invocation of SLAC's `environ` shell script, i.e., a line near the beginning of the file that looks something like this:

```
eval '/usr/local/bin/environ /bin/csh -i${?prompt} -e emacs:vi'
```

The string "`/bin/csh`" must be changed to correspond to your new login shell; e.g., if you've just changed to the tcsh shell, change this line to:

```
eval '/usr/local/bin/environ /bin/tcsh -i${?prompt} -e emacs:vi'
```

Save the file and exit the editor, then logout and login again.

WARNING!! Although new accounts are set up with additional dot files (e.g., `.profile`) that are used by some other shells at login time, these are intended as examples only and are not supported by SCS. Note also that the full path for the Bourne Again shell is /usr/local/bin/bash, not /bin/bash.

Customizing the Session: Defining the Shell Environment

UNIX allows you to customize your interaction with the shell program in many ways. You can create abbreviations for files or directories or lengthy commands; determine the directories the shell uses to search for the programs and commands to run; create the prompt the shell uses when it interacts with you; and have UNIX set your terminal type every time you login, without having to type the command yourself.

Similar to program variables (place holders of information), shell and environment variables store values that the shell uses in executing your commands. Shell variables help determine how the shell program interacts with you. You can create your own shell variables and assign them values with the **set** command, and you can insert the variable value in a command line by prefixing the variable name with $.

`$ set docdir=/usr/local/doc`	Sets the variable **docdir** to `/usr/local/doc`. That is, **docdir** becomes an abbreviation for `/usr/local/doc`.
`$ echo $docdir`	Shows the value assigned to **docdir**.
`/usr/local/doc`	
`$ cd $docdir`	The current working directory changes to `/usr/local/doc`.
`$ unset docdir`	Removes the shell variable **docdir**. Now, you can no longer use **$docdir** to refer to the directory `/usr/local/doc`.

The procedure shown in the preceding sample session sets the shell variable for use during your present interaction with the current shell. For example, if you used the **script** command, a new shell would begin, and this shell variable would no longer be set. Even if you used only the login shell, the next time you login you would have to repeat the procedure of setting the shell variable.

If you want to use the shell variable every time you interact with the C shell, use a text editor to add the command line to the `.cshrc` file. (See "Editors" on page 39 for more information on editors.) The `.cshrc` is the **C Sh**ell **R**un **C**ommands file, a dot file in your home directory. (See "Working With Files" on page 15 for more information on dot files.) Every new UNIX account gets a `.cshrc` file that sets a number of defaults such as the architecture of the host machine and via SLAC's `environ` shell script. The `environ` script is invoked via an `eval` command that looks like this:

```
eval '/usr/local/bin/environ /bin/csh -i${?prompt} -e emacs:vi'
```

Changes should normally be added after the invocation of the `environ` script.

Use the **set** command to see shell variables and their values.

`$ set`	Without argument, **set** displays shell variables and their values.
`argv ()`	
`cwd /u/sf/ilse`	
`term vt100`	
`history 50`	

A few shell variables have special meaning to the shell program. By assigning values to these variables, you customize the manner in which the shell executes your commands. The one covered here is the **path** variable. **path**, as well as others such as **prompt**, **history**, and **term**, are set in your default `.cshrc` file, and, of course, you can change them if you like.

About Path and Environment Variables

One of the most important shell variables is the **path** shell variable. The value for the **path** shell variable is a list of directories. This list of directories is called your *search path*. UNIX searches these directories when it looks for a program or command specified on the command line. If a program is in your search path, you can just type the name of the program; you don't have to type its absolute or relative pathname. Suppose your search path is:

```
/u/<gg>/<userid>/bin /usr/local/bin /usr/afsws/bin /usr/ucb /bin /usr/bin
```

where"`/u/<gg>/<userid>/bin`" identifies your personal bin directory. Whenever you type a program name or command, the shell looks first in your bin directory, second in the directory /usr/local/bin, third in the directory /usr/afsws/bin, and so on. It keeps looking until it either finds the program or command with that name or finishes looking through all the directories in your search path.

$ **joes_program**	Let's say you execute `joes_program` frequently.
joes_program: command not found	The command is not in the search path.
$ **set path=($path /joe/bin)**	Adds the directory where `joes_program` is located to the search path for that particular session only.

The procedure in the sample session above adds a directory to your search path for your present session with the shell. Notice the following features of the above **`set path`** command:

- Parentheses must surround the value to the right of the equals sign because it is a blank-separated list of words rather than a single word or quoted string.
- The current value of your **path** shell variable can be referenced on the right of the equals sign; this makes it easy to add directories to the end (as in this example) or the beginning of your search path.

Your search path may include a period to represent your current working directory. Though this is sometimes convenient, it presents a significant security risk via so-called "trojan horse" attacks, particularly when the period is at the beginning of the search path.

WARNING!! At present, SLAC's `environ` script inserts a period at the beginning of your search path; however this policy is currently under review and may be changed in the near future.

If there is a period early in your search path, please consider moving it to the very end or, better yet, removing it altogether. It is always possible to execute a command in your current directory simply by prefixing it with "`./`".

Removing a directory from your path shell variable is a little less straightforward than adding one. Here's an example that will safely remove the period representing your current directory from anywhere in your search path (note that the blanks in this example are significant):

```
set path=( ‘echo " $path " | /bin/sed -e "s% . % %g"‘ )
```

If you find that you frequently use a program or command not included in your default search path, you can edit or add a "**`set path=`**" command in your `.cshrc` file.

After you add or modify command lines in the `.cshrc` file, the changes will be in effect for all subsequently started shells. However, neither your currently running shell, nor any of its ancestors will automatically see the changes. To invoke the commands added to your `.cshrc` file in your current shell, use the **source** command, as shown in the

following sample session.

$ **source .cshrc**	Now the commands in the `.cshrc` file run, so the shell variables are set as you specified in the `.cshrc` file.

Environment variables are similar to shell variables, but other programs besides the shell use their values. Typically, these environment variable names are all uppercase. An example of an environment variable is PRINTER. The value associated with this environment variable determines the default printer. At SLAC a variety of printers are available to UNIX users. They are listed in the file `/etc/qconfig` (RS/6000 users only) or `/etc/printcap` (all other UNIX users). The following sample printer description from the former file shows an HP printer located in the computer building:

```
...
* Computer Building, 1st Floor, George Maclin
hpcgb1a:
        host = lpd01
        s_statfilter = /usr/lpd/bsdshort
        l_statfilter = /usr/lpd/bsdlong
        rq = hpcgb1a
        device = rlp218
...
```

The same printer in the latter file appears as:

```
...
hpcgb1a|Computer Building, 1st Floor, George Maclin:\
        :sh:\
        :mx#0:\
        :rm=lpd01:\
        :lp=:\
        :sd=/usr/spool/lp/hpcgb1a:\
        :rp=hpcgb1a:
...
```

For more information about printing, see "Printing" on page 43.

The commands to set, unset, and look at environment variables are slightly different than those used with shell variables. As for shell variables, command lines to set environment variables can be added to your `.cshrc` files.

```
$ setenv PRINTER hpcgb1a
```

Sets environment variable PRINTER to hpcgb1a.

```
$ unsetenv PRINTER
```

The environment variable PRINTER is no longer set.

```
$ printenv
```

Prints environment variables and their values.

```
HOME=/u/sf/ilse

SHELL=/bin/tcsh

TERM=vt100

USER=ilse

...
```

Creating Shorthand Names for Commands: Aliases

Aliases allow you to create shorthand names for frequently used or lengthy command lines. When the alias appears in the command line that the shell reads, its text is replaced by the definition of the alias. Use the **alias** command to create aliases and the **unalias** command to delete aliases.

```
$ alias prt lpr -Phpcgb1a
```
prt will be shorthand for printing on the `hpcgb1a` printer.

```
$ alias prt
lpr -Phpcgb1a
```
Shows commands associated with **prt**.

```
$ alias
ll      ls -l !*
ls      (ls -F)
prt lpr -Phpcgb1a
...
```
By itself, **alias** displays the aliases you set up.

```
$ prt unix.doc
```
Prints file `unix.doc` on the `hpcgb1a` printer.

```
$ unalias prt
```
Removes the alias **prt**.

This sample session sets up the alias for your use during the present interaction with the current shell. However, if you know you want to use the alias in later interactions with the shell program, you should add the command line to the `.cshrc` file in your home directory.

Note: *The example above shows another way to produce the same effect as the command* **`setenv PRINTER`**. *The environment variable is discussed in "About Path and Environment Variables" on page 32*

Redirecting Input/Output

Normally, UNIX commands needing user input require you to type the input at the keyboard. Most UNIX commands show you the output of the command by displaying it on your terminal screen. The keyboard is the standard input and the terminal is the standard output. But often you might like to store the lengthy output of a command in a file or cause the input for a command to come from a specific file. The shell provides a means for redirecting where a command sends its output or receives its input. The characters <, > and >> are used to redirect input and output**.**

```
$ who
raines  ttyp0  Aug 20 09:08  (CGIBM1.SLAC.Stan)
ohnishi ttyp1  Aug 20 09:17  (134.79.128.89:0.)
meb     ttyp3  Aug 20 08:15  (sisyphus:0.0)
```
who shows who is logged on to the system

```
$ who > who.doc
```
Places the output of the **who** command into the file `who.doc`. If the file already exists, its contents are replaced by the output of the **who** command. If the file does not already exist, it is created.

```
$ sort who.doc
```

sort re-orders the contents of file `who.doc` alphabetically, and writes the result to standard output.

```
meb      ttyp3   Aug 20 08:15  (sisyphus:0.0)
ohnishi  ttyp1   Aug 20 09:17  (134.79.128.89:0.)
raines   ttyp0   Aug 20 09:08  (CGIBM1.SLAC.Stan)
```

WARNING!! In the sample session above, the output of* `who` *was placed in the* `who.doc` *file. When you redirect the output of a command, you can either make up a new file to store the output or use a file that already exists. If you use a file that already exists, however, the contents of that file will be replaced by the redirected output. To avoid accidentally overwriting an existing file with the > character, use a text editor to put the command* `set noclobber` *in your* `.cshrc` *file.

```
$ who >> info.doc
```

Append the output of the **who** command to the end of file `info.doc`. The contents of this file remain intact with the output of the **who** command attached at the end.

The **tr** command in the following example only reads standard input and writes standard output unlike some commands which allow you to specify an alternative input and/or output file.

```
$ tr a-z A-Z < Text > Text.upper
```

Translates all lowercase characters in the file `Text` to uppercase in the file `Text.upper`.

The **cat** command allows you to concatenate files together. It simply copies the files you specify as arguments to its standard output, which defaults to your terminal screen. With the > character, however, you can redirect the output from the screen to another file. Note the warning in the paragraph above.

```
$ cat sec1.doc sec2.doc > unixbook
```

Strings together `sec1.doc` and `sec2.doc`. Redirects the output of the cat command to the file `unixbook`.

Often it is convenient to make the output of one command become the input of another command. The UNIX shell provides a means of connecting the output and input of two commands via a mechanism called a *pipe*. The pipe character (|) is placed between two commands when the first command's output should be the input of the second.

```
$ ls -l /bin  | more
```

Lists the files in directory `/bin` (the command **`ls -l /bin`**), one screenful at a time (the command **`more`**).

```
total 2722
-rwxr-xr-x  2 root        4008 Jul 21  1992 [*
-r-xr-xr-x  1 root        9080 Jul 21  1992 ar*
-r-xr-xr-x  1 root      122880 Jul 21  1992 as*
...
```

The characters in the command line that redirect input/output (<, >, and >>) and pipe commands (|) are not part of the commands themselves. Rather, they are special characters, which hold special meaning to the shell. That is, the commands themselves know nothing about the <, >, and | characters, but the shell recognizes them and provides the mechanism by which this redirection takes place.

Referring to Groups of Files

The shell recognizes other special characters besides those that redirect input/output and serve as pipes between commands. The most common are those associated with filenames. When included in a file's name, these characters allow you to specify groups of files more easily. For more information about using these characters, give the command **man csh**.

TABLE 4. Special Characters Used with Filenames

The Character	What It Means
*	Refers to any string of zero or more characters.
?	Refers to any single character.
~	Has special meaning when associated with a username as in `~smith`. In these cases, this refers to the user smith's home directory. Thus, `~smith/.login` refers to the file `.login` in smith's home directory. By itself, ~ refers to your home directory.
[ ]	Indicates any single character of the ones in the brackets.

```
$ ls                                        Lists all files in the working directory.
Makefile    main.c    queues.h  sub.c
globals.h  queues.c  sim.h     test1
$ ls *.c                                    Lists all files ending with .c.
main.c     queues.c   sub.c

$ ls s*.c                                   Lists all files beginning with s and ending with .c.
sub.c
$ ls test?                                  Lists all files beginning with test, ending in any
                                            single character.
test1
$ ls *.[ch]                                 Lists all the files, ending in .c or .h.
globals.h       queues.c        sim.h
main.c          queues.h        sub.c
$ ls ~smith                                 Lists all files in the home directory for smith.
Work  mbox  bin
```

Name Completion

A very useful feature of tcsh is name completion, that is, the ability of the shell to complete the typing of a name on the command line, given a unique abbreviation. This works for command names, filenames, references to shell and environment variable names (i.e., when the name follows a $-sign), and the ~username convention. It is triggered by typing part of the name and pressing the TAB key. If the part of the name you type does not uniquely identify a complete name within the appropriate class of names, any additional unambiguous characters are added to what you typed and the terminal bell is rung. At this point you can type CTRL-d to display a list of matching names. You can then type a few more characters and try pressing TAB again. There are a number of ways in which the behavior of this feature can be modified (the above behavior is the default); for details, enter "**`man tcsh`**" .

The C shell has a similar but more limited feature. It only works with filenames and the ~username convention, is triggered by the ESC key rather than the TAB key, and is disabled by default (to enable it, define the special shell variable "`filec`" to any value in your `.cshrc` file, e.g., by adding a line like "`set filec=on`").

Keeping Command History

The shell has a mechanism that allows you to review the commands you entered most recently. The number of commands you can review is determined by the value of the **history** shell variable which is 50 as set in your default `.cshrc` file. You can then use a variety of commands to review and repeat your saved commands:

`$ set history=5`	Sets number of commands for UNIX to save at n (in this case, 5).
`$ history`	Displays the previous 5 commands, as set in the step above.
`15 who`	
`16 finger smith`	
`17 talk smith`	
`18 cd ~`	
`19 rm who.doc * [.1in]`	
`$ !!`	Repeats the immediately preceding command.
`history`	
`16 finger smith`	
`17 ...`	
`:`	
`$ !finger`	Repeats most recent command starting with string finger.
`finger smith`	
`$ !16`	Repeats command 16.
`finger smith`	

When using tcsh (but not csh) you can scroll through your command history using the up- and down-arrow keys, position the cursor within a command using the left- and right-arrow keys, and edit the command before reexecuting it.

Sometimes you may want to save your commands and the shell's response in a file. That way you can see or print the

information later. You may want to record the running of a specific program and inspect this session later. The **script** command saves the record of your interaction with the shell in a file named `typescript` in the working directory. You must give the **script** command before you start the process you wish to record. You can specify different file names if desired by typing **script filename**; type **man script** at the shell prompt for more information. To signal the end of the recording, type the command **exit**. At that point, you can look at the `typescript` file by typing **more typescript** at the prompt.

```
$ script                                          Saves all commands and shell
                                                  responses that follow.
Script started, file is typescript
csh> testprog
What is the result of 3+4? 7
Correct!
csh> exit                                         Ends the recording.
csh> Script done, file is typescript
$ more typescript                                 Displays the commands and shell
                                                  responses just saved.
Script started on Tue Nov 06 23:57:06 1990
csh> testprog
What is the result of 3+4? 7
Correct!
csh> exit
script done on Tue Nov 06 23:57:06 1990
$
```

CHAPTER 5

Editors

There are a number of editors available in UNIX, including emacs, vi, and uni-xedit. You are free to choose any of these or another. No specific preference is given.

GNU Emacs

GNU Emacs is arguably the most powerful and complex of the editors, constituting practically an entire computing environment within itself. Modern versions of Emacs have menu and mouse support if invoked on an X-Terminal, easing the learning curve for this editor.

GNU Emacs functions do things like move one line down, delete a word, move the cursor to the end of the file. Most of the useful functions are bound to particular key strokes so that you can invoke a function by hitting two or three keys. A typical key stroke involves the CTRL or the ESC key and another key. In this section, ^d means holding down the CTRL key while you press the key d; ESC-d means pressing the ESC key and then pressing the key d (not simultaneously). The most common functions are available in the menus in the X Windows version.

For more detailed instruction on the GNU Emacs editor, get a copy of the *GNU Emacs Manual* by Richard Stallman (about 300 pages) from the Help Desk in the Computer Building Lobby. You can access the man page with the command **`man emacs`**.

TABLE 5. How to access the Emacs Editor

Function	Prompt User Sees	What User Types
Enter editor	$	**`emacs`** RETURN
Exit editor		**`^x^c`** (If the file has been changed, Emacs will prompt whether file is to be saved before exiting.)
	$	

You can access an online, interactive Emacs tutorial by typing **`emacs`** and following the instructions.

Vi

vi (pronounced vee-eye) is a dual mode-oriented editor. One mode allows text entry by simply typing (called text entry mode); the other allows text manipulation by using the typing keys (called command mode). This is a completely different approach to editing than used by most other text editors and word processors. It provides some incredible possibilities for speed because one does not have to look to find special function keys to perform certain editing tasks. Also, many more commands can be placed within reach of one's hands because all of the keys on the keyboard are available for use during command mode operations.

TABLE 6. How to Access the vi Editor

Function	Prompt User Sees	What User Types
Enter editor	$	**`vi`** *filename* RETURN
Exit editor		**`:wq`** (write/quit) RETURN or **`:q`** RETURN (if no changes have been made.) or **`:q!`** RETURN (quit without saving changes)
	$	

You can access an online vi tutorial by editing the file **`/usr/local/doc/vitutor`** and the man page with the command **`man vi`**. No other local documentation is available, although there are books such as *Learning the Vi Editor* by Linda Lamb, published by O'Reilly and Associates, Inc. (Nutshell Handbook) on the market.

Uni-Xedit

Uni-Xedit is an editor that SLAC has licensed to ease the transition from VM to UNIX for those who are familiar with XEDIT on VM. It is available on the Suns and RS/6000s onsite. But, you may want to begin learning one of the native UNIX editors since those will be more generally available on all UNIX machines you will encounter.

TABLE 7. How to Access the Uni-Xedit Editor

Function	Prompt User Sees	What User Types
Enter editor	$	**`xee`** *filename* RETURN or for an X-Windows version: **xeg** *filename* RETURN

TABLE 7. How to Access the Uni-Xedit Editor

Function	Prompt User Sees	What User Types
Exit editor		**`file`** RETURN or **`quit`** RETURN (if no changes have been made.) or **`qquit`** RETURN (quit without saving changes)
	$	

Other Editors

There are many other editors available, each of which has its adherents. Since choice of editors is a matter of personal preference, we list them here. However, there is no support from SCS for any of these editors, and you may have difficulty in getting help in their use.

TABLE 8. Other Editors at SLAC

nedit	A GUI editor developed by Fermilab that has some resemblence to popular Mac and PC editors. Almost entirely mouse- and men- based. Type **`man nedit`** to get started.
ne	A full-screen character editor resembling some PC editors. Has menus but depends on control keys for most tasks.
xedit	A simple mouse-based editor for X (not a VM XEDIT clone). Type **`man xedit`** to get started.
axe	Another mouse-based editor for X, more sophisticated than xedit. Type **`man axe`** to get started.

CHAPTER 6

Printing

SLAC has a large number of printers from a variety of vendors that can print text, PostScript, and graphics in black and white and color on paper and foils. In this section you will learn:

- How to change the default printer on a new UNIX account. Note that this **must** be done because the default setting will not work.
- How to find a printer near you.
- What the commonly used print commands are, what they can be used for, and how to get more information about them.
- How to use printing command options such as duplex, double-column, landscape mode.
- How to cancel a print job from the local printer queue.
- Tips on what to do when your output doesn't show up.

Note: *Many SLAC printers, in particular Imagen printers, can only print text and are not PostScript-compatible. A PostScript file sent to a non-PostScript printer will not produce the desired output, may take a long time to print, and may waste a lot of paper. By UNIX convention, names of PostScript files end in ".ps" or ".eps". Such files should be printed on PostScript printers.*

Changing the Default Printer on New UNIX Accounts

Because printers are distributed across SLAC, no single printer serves a majority of UNIX users. Therefore, for new UNIX accounts the default printer has been set to **InvalidPrinter**. If you have a new account, first change **InvalidPrinter** in your `.cshrc` file to the name of a printer that makes sense for you. If you don't, you will receive an error message, like those given below, when you try to print. We suggest that you ask someone in your group to recommend the name of a printer, or you may be able to find one in the file `/etc/qconfig` or `/etc/printcap` (see section "Finding the Names of Printers" on page 44). To change the default printer:

1. Edit your `.cshrc` file (e.g., with the emacs editor).
2. In the line:

   setenv PRINTER InvalidPrinter

   change **InvalidPrinter** to a printer of your choice, e.g., `hpcgb3b.`
3. Save the file.

Note: *The change will take effect when the.cshrc file is executed, for example, by opening a new shell window. The simplest way to ensure that it is executed might be to logout and then login.*

The **setenv** command in your `.cshrc` file is preceded by this explanatory comment:

```
# Set the value of the PRINTER environmental variable to the name
# of your default printer. For printer names, see the file
# /etc/qconfig on AIX systems or /etc/printcap on other systems).
# The default value, "InvalidPrinter", prevents the lp command from
# accidently submitting print jobs to the batch queue.
```

The next section discusses the contents of `/etc/qconfig` and /`etc/printcap`.

If you try to print a file to **InvalidPrinter**, the response will depend on the system on which you issued the command (e.g., RS600, Sun) and the print command you used (e.g., lpr, enscript). For example:

Sun responses:

```
ilse@scssun1 $ lpr my-aid-file
lpr: InvalidPrinter: unknown printer

ilse@scssun1 $ enscript my-aid-file
enscript: 4 lines were wrapped because of length.
[ 2 pages * 1 copy ] spooled to InvalidPrinter
lpr: InvalidPrinter: unknown printer
```

RS6000 responses:

```
ilse@sisyphus $ lpr my-aid-file
enq: (FATAL ERROR): 0781-050 Bad PRINTER or LPDEST env. variable InvalidPrinter.

ilse@sisyphus $ enscript my-aid-file
enscript: 4 lines were wrapped because of length.
[ 2 pages * 1 copy ] spooled to InvalidPrinter
enq: (FATAL ERROR): 0781-048 Bad queue or device name: InvalidPrinter.
```

Changing the Default Printer for the Current Shell Window

To change the default printer for the current shell window, enter the command:

$ setenv PRINTER *printer-name*

substitute the name of the chosen printer for *printer-name*. For example, **setenv PRINTER imcgb3b**.

This command changes the default printer until you end that shell window. In any other shell window (including new ones) the default printer will be in effect.

Finding the Names of Printers

Either open the file `/etc/qconfig` on AIX (RS6000) system or file /`etc/printcap` on other systems in an editor or print it in a shell window. For example, the command:

more /etc/qconfig

will list one screenful of information of the file at a time. (Press the space bar to get the next screenful.) Note that both files are quite long.

The following is a sample description from the file `/etc/qconfig` for an Apple LaserWriter printer named `lwcgb3a.`:

```
* Computer Center Comp Group lounge
lwcgb3a:
        host = lpd01
        s_statfilter = /usr/lpd/bsdshort
        l_statfilter = /usr/lpd/bsdlong
        rq = lwcgb3a
        device = rlp86
rlp86:
        backend = /usr/lpd/rembak

lwcgb3a-ps:
        host = lpd01
        s_statfilter = /usr/lpd/bsdshort
        l_statfilter = /usr/lpd/bsdlong
        rq = lwcgb3a
        device = rlp87
rlp87:
        backend = /usr/lpd/rembak

lwcgb3a-tx:
        host = lpd01
        s_statfilter = /usr/lpd/bsdshort
        l_statfilter = /usr/lpd/bsdlong
        rq = lwcgb3a
        device = rlp88
rlp88:
     backend = /usr/lpd/rembak
```

The most useful information for most new users is:

- the comment (in the above example, `* Computer Center.....`) that indicates the location of the printer. Sometimes the name of person whom you can contact for information is given, and
- the name of the printer given directly beneath the comment (in the example the printer name is `lwcgb3a).` Aliases for this printer are `lwcgb3a-ps` and `lwcgb3a-tx`.

Many printer names give a clue about the printer's manufacturer and location; in this example, an Apple Laserwriter located in the Computer Group Building on the third floor.

The same printer is described in `/etc/printcap` as follows:

```
lwcgb3a|lwcgb3a.ps|lwcgb3a.tx|wingz2|lwcgb3a|Computer Center Comp Group
lounge:\
        :sb:\
        :sh:\
        :mx#0:\
        :lp=:\
        :rm=lpd01:\
```

```
:rp=lwcgb3a:\
:sd=/usr/spool/lp/lwcgb3a:
```

Aliases for the printer name `lwcgb3a` are given in the first line after the first "|" character. The `:ty` field indicates the printer model.

The commands **man qconfig** and **man printcap** will describe the printer description fields in more detail.

Printing Files and Other Text

Print Commands

Commands you can use for printing include **lpr, enscript,** and **pslpr**. Which of these you select depends on your file, your print requirements, and your choice of printer. Remember, though, that some printers at SLAC can only print text (e.g., the Imagen Pprinters). For full reference information about these commands, enter the **man** command; for example, **man lpr**.

Two other commands you will likely find useful are:

- **lpq** to display the print queue for the backlog of jobs waiting to be printed.
- **lprm** to cancel a print job.

Print command	Description
lpr	Use it for printing text files. It also prints TeX, troff, ditroff, PostScript files and standard plot data to the specified or default printer.
enscript	Turns a text file into a PostScript file. Particularly useful features include: multi-column, default and tailored header line on each page, portrait and landscape modes, printer-dependent options such as duplex, paper tray. See examples in section "Specifying How the File Should Print" on page 47.
pslpr	Prints PostScript files with similar options as **enscript**. See examples in section "Specifying How the File Should Print" on page 47.
qmslpr	Prints files on QMS printers only. Similar to lpr, but has options for duplex, portrait/landscape mode, multiple copies (collated and uncollated), input and output bin specifications, and others.
lpq	Displays the print queue for the backlog of jobs waiting to be printed.
lprm	Removes or cancels the job queued in the local print queue of the computer on which you executed the command. It does **not** cancel the job after it has been removed from the local print queue by a printer server (which is another computer). For examples see section "Locating and Canceling Print Jobs in the Print Queue" on page 49.

Basic Print Command Examples

Following are examples of the **lpr** (print) and **echo $PRINTER** (display default printer) commands given on an RS6000. The default printer was changed from **InvalidPrinter** to **hpcgb3b**, an HP Laserjet 4si mx printer. Responses will vary according to the platform (e.g., RS600 or Sun) on which you give these commands and according to the printer you query (e.g., Apple Laserwriter, Hewlett Packard printer, etc.).

Examples of *-Pname*, the name of the printer on which you want to print, and other useful print options are given in the next section.

Jobs will be printed on the default printer, unless you specify otherwise in the argument.

WARNING!!
If you have a new UNIX account, be sure to change the default printer from InvalidPrinter to the name of the printer that you wish to use in your `.cshrc` file (see section "Changing the Default Printer on New UNIX Accounts" on page 43).

ilse@sisyphus $ **echo $PRINTER** displays default printer name; in this case, hpcgb3b

```
hpcgb3b
```

$ **lpr chap2.ps** prints file chap2.ps on the default printer

$ **lpq** queries the print queue for the default printer; job number is 30

```
Queue      Dev      Status      Job  Files      User   PP  %   Blks  Cp  Rnk
hpcgb3b   rlp22     READY
hpcgb3b   hpcgb3b is ready and printing
hpcgb3b   hpcgb3b   RUNNING     30   chap2.ps   ilse   0   0   217   1   0
```

Specifying How the File Should Print

Some printers (e.g., HP Laserjet 4si mx) can print duplex (that is, both sides of the paper), manual feed, and select paper from one of two paper trays.[1] **enscript** for text files and **pslpr** for PostScript files let you control these printer features with the **-S** option (see examples below). The catch is that the features you select must match a keyword in the PostScript Printer Description (PPD) file for the printer of your choice. PPD files, which are supplied by the printer manufacturer, reside in `/usr/local/lib.shared/ts/ppd`, and they are not easy to read. If you can't make sense out of them, see if someone more knowledgeable in your group can help you.

1. The examples shown in this section work on an HP Laserjet 4si mx printer. They will most likely work on other HP printers that have multiple input trays and are capable of printing duplex. The examples may not produce the same result on printers by other vendors.

To:	Give command:	Comment
Print header (filename, current date, & page number) on each page of text file	**enscript -Phpcgb3b mydoc**	`mydoc` is filename of a text file; `hpcgb3b` is printer name. Omit **-P***printer-name* if you want to use the default printer.
Print 2-up landscape	**enscript -2r -Phpcgb3b mydoc**	`mydoc` is filename of a text file; `hpcgb3b` is printer name. Omit **-P***printer-name* if you want to use default printer.
Print 2 copies of a file	**enscript -#2 -Phpcgb3b mydoc**	`mydoc` is filename of a text file; `hpcgb3b` is printer name; #2 specifies 2 copies. Omit **-P***printer-name* if you want to use default printer.
Turn duplex off	**pslpr -Phpcgb3b -SDuplex=Off mydoc**	`mydoc` is the filename of a PostScript file. **Duplex=On** is the default for printer hpcgb3b. Same options work for **enscript**.
Print from the lower paper tray	**enscript -Phpcgb3b -SInputSlot=Lower mydoc**	Printer `hpcgb3b` has two trays; the upper tray is used by default. Same options work for **pslpr**.
Turn duplex off and print from the lower tray	**enscript -Phpcgb3b -SInputSlot=Lower -SDuplex=Off mydoc**	Combination of the options of two examples above. Same options work for **pslpr**.
Print duplexed pages so that they must be turned over from the right side	**pslpr -Phpcgb3b -SDuplex=DuplexNoTumble mydoc**	**-SDuplex=DuplexTumble** specifies duplexed pages must be turned over from the bottom of the page. `mydoc` must be a PostScript file. Same options work for **enscript**.
Print several page ranges in a PostScript file	**pslpr -Phpcgb3b -i3-7 -i10-15 mydoc.ps**	Prints pages 3 through 7 and 10 through 15 of file `mydoc.ps` on printer `hpcgb3b`.
Print PostScript file in landscape mode	**pslpr -Phpcgb3b -L mydoc.ps**	Omit **-Phpcgb3b** to print on default printer.
Print text file in two columns in landscape mode	**enscript -2r -Phpcgb3b mydoc**	Omit **-Phpcgb3b** to print on default printer.

To:	Give command:	Comment
Print two collated copies of file in landscape mode on print qmscgb3a	**qmspr -ls -#2 -c -Pqmscgb3a my-aid-file**	Specify **-l**language for the language to be used by printer to interpret the document. **-ls** is landscape; **#2** requests two copies; **-c** asks for copies to be collated; printer to be used is **qmscgb3a**; file **my-aid-fil**e is a text file.

Locating and Canceling Print Jobs in the Print Queue

Locating a Print Job in the Print Queue

When you issue a print command, a copy of the file in made in the local print queue, that is, the print queue of the machine on which the print job originated. From there it is sent to the remote print server queue. The remote print server is identified by the `:rm` tag in file `/etc/printcap` and the `:host` tag in file `/etc/qconfig` for the printer on which the file is to be printed.

When issued on a Sun workstation, the command **lpq** displays print jobs in the local print queue; when issued on an RS/6000, the command displays jobs in the local queue and the remote print server queue. Be aware that if the printer is lightly loaded, the print job will move so quickly from the local queue to the remote server queue to the printer that by the time you issue the **lpq** command, the job may no longer be in the queue. The **-l** (long) option on the **lpq** command will give the name of the host on which the print job originated.

$ **lpq** query the print queue for the default printer

```
Queue    Dev     Status    Job  Files        User    PP   %    Blks    Cp    Rnk
hpcgb3b  rlp22   READY
hpcgb3b hpcgb3b is ready and printing
hpcgb3b hpcgb3b  RUNNING   30   chap2.ps     ilse    0    0    217     1     0
```

Cancelling a Print Job

You can only cancel a print job that is in the local printer queue; you cannot remove it from the printer server queue. Use the **lprm** command with either the job number displayed by the **lpq** command or your userid (see examples below). If you specify your userid, all your jobs currently in the local queue will be removed. Note that you cannot remove jobs that belong to someone else.

$ **lpr chap2.ps** print file chap2.ps on the default printer

$ **lpq** query the print queue for the default printer; the job number is 30

```
Queue        Dev           Status          Job    Files              User
hpcgb3b      rlp22         READY
hpcgb3b      hpcgb3b is ready and printing
```

```
hpcgb3b      hpcgb3b       RUNNING         30     chap2.ps           ilse
```

```
ilse@sisyphus  $ lprm 30                                    remove job number 30
$ lpd01: dfA030sisyphus dequeued
lpd01: cfA030SISYPHUS.SLAC.Stanford.EDU dequeued
```

```
ilse@scssun1  $ pslpr ~ilse/phonebook-ps/chap2.ps           print PostScript file ~ilse/phone-
                                                            book-ps/chap2.ps on default printer
ilse@scssun1 $ lpq                                          query local job queue for default printer; the
                                                            job number is 33
no entries
scssun1: sending to lpd01
Rank   Owner      Job  Files                          Total Size
1st    ilse       33   standard input                 110921 bytes
```

```
ilse@scssun1 $ lprm ilse                                    remove all of user ilse's jobs from job queue
dfA033scssun1 dequeued
cfA033scssun1 dequeued
```

A job that is printing cannot be cancelled with the **lprm** command. You may be able to stop it at the printer by taking the printer offline or pressing the reset button. You will need to check the documentation for the printer in question.

Printing Problems

Print Job Too Large

When you give a print command, a temporary copy of the print file is made in the spool area of the machine on which you are running. If there is not enough space for the file, you will receive an email message from lpd, subject: printer job, with text saying: Your printer job could not be printed. This message, as unhelpful as it is, might indicate that spool space was insufficient, especially if your file is quite large. To get around the problem, create a symbolic link to your print file from the spool space with the command:

lpr -s -P*printername filename*

No temporary copy of the file will be made; therefore.....

WARNING!! ***Do not erase or change the file until the file has been printed.***

No Output is Produced by Print Command

You've given a print command, waited for an appropriate length of time for the output to show up, but the printer output tray is empty. What to do?

1. Check that the printer is operational: Is it in print ready status? Does it have adequate paper in the input paper tray?

2. If the answer is yes, give the command:

   lpq -P*printer-name*

   to see if your job is sitting in the print queue of the machine on which you gave the command. (You can omit the **-P** option if you printed to the default printer.) If it is and the printer is operational, the print queue may need to be restarted. If you gave the print command on a Sun workstation, enter the command:

   lpc restart -*printername*

   Note: *The lpc command will only work on a Sun, not an RS6000.*

3. If you are printing from an RS6000 and the print queue appears to be stuck, call the SCS Help Desk at x4357. A UNIX administrator with special systems privileges will need to restart the print queue.

Need Help?

Try the following sources:

- Check the man pages for the print commands discussed in this chapter; for example, give the command **man lpr**.
- Talk to someone more knowledgeable in your group.
- Call the SCS Help Desk at x4357 or send email to *help@slac.stanford.edu*.
- Send email to *unix-admin@slac.stanford.edu*. Someone in the SCS UNIX group will respond to you.

CHAPTER 7

Communicating With Other Users

On SLAC's UNIX system, most communication between users occurs with electronic mail and via Network News (Netnews), a computer-based conferencing system and bulletin board that is available on SLAC's network. Elm is the supported e-mail system on UNIX; on Macs and Windows NT the supported system is Eudora-Pro. Although there are several news readers available, SCS does not support one.

Using E-mail

Elm

The supported e-mail system on UNIX is elm, which is fairly intuitive. For documentation see the README file in the subdirectory */usr/local/doc/Reference/elm/elm-2.4* or enter the command **man elm**. Within elm, enter **?** to receive help. We suggest that you familiarize yourself with a UNIX editor, such as emacs or vi, if you aren't already before using an editor within elm. The default editor in elm is emacs; if emacs doesn't exist on UNIX of the machine you are using, the default is vi. Note that elm uses the default printer for printing, unless you change the default. For more information, see "Changing the Default Printer on New UNIX Accounts" on page 43.

Your initial session as a new elm user might look something like the session below.

```
vanilla@sisyphus $ elm

Notice:
This version of ELM requires the use of a .elm directory in your home
directory to store your elmrc and alias files. Shall I create the
directory .elm for you and set it up (y/n/q)? n


    Mailbox is '/usr/spool/mail/vanilla' with 2 messages [ELM 2.4 PL25]
```

```
   1   Jul 5  meb@mailbox.SLAC.S (17)   test
   2   Jul 1  meb@mailbox.SLAC.S (38)   Re: mail/pine problems for user hebe




 You can use any of the following commands by pressing the first character;
 d)elete or u)ndelete mail,  m)ail a message,  r)eply or f)orward mail,  q)uit
    To read a message, press <return>.  j = move down, k = move up, ? = help

Command:
```

FIGURE 5. Sample elm E-mail Session

Note: *When you enter an elm command, such as "m" for mail message, do not to press the ENTER key. As soon as you type "m", elm executes the command; pressing ENTER will likely cause a second command to be executed*

To display a list of elm commands and how to invoke them, enter **?** and then enter **?** again in response to the prompt.

Finding Email Addresses

An easy way to locate an email dress, phone number, office location, etc. for people at SLAC is using the Phonebook button on the WWW SLAC Home pages. It will display a search form on which you can enter search data. To locate email addresses of people in the physics community world-wide use the resources listed under the heading *Information from SLAC* on the WWW Highlighted or Detailed Home page. For more information about WWW, see section "World Wide Web (WWW)" on page 10.

NetNews

Through NetNews you have access to many news groups that speak to a variety of subjects. UNIX users (and other SLAC computer users on other platforms) can share information and publicize upcoming events using local SLAC Netnews groups and access information from all over the world.

Netnews has many similarities to a normal electronic mail delivery system, but also some major differences. A normal e-mail system provides person-to-person communications, a transaction between sender and recipient in which a third party cannot view the mail contents. An e-mail system can typically be extended by using mailing lists through which each mailing list member receives a copy of the mail. Bulletin boards extend the e-mail model by allowing all bulletin board subscribers to view all postings to the board. Usually a bulletin board resides on one computer to which subscribers log on. In contrast, the Netnews "board" floats around the world travelling from one computer on the network to the next. Since new messages can be entered at any computer, the "board" will usually look slightly different at every site.

For newsgroups local to SLAC see newsgroups with names starting with slac. You might, for example, want to periodically read the entries in slac.users.unix.

Currently there is no single supported newsreader for UNIX. SCS is presently evaluating several news support models whose goal is to provide reliable news service using news readers on all supported platforms. IBM AIX/6000 and Sun users have access to the X-windows application **xrn**, which is also available in line-mode as the command rn. **xrn** is a point-and-click interface. For more information about the line-mode rn enter the command **`man rn`**. To use **xrn** you must first start an X-windows system; for information, see "Workstations and Window Systems" on page 8.

If you use the editor emacs, consider using the emacs newsreader. It has both a line mode and X-window interface. To use the X-windows interface:

1. Start X-windows.
2. Enter the command: **`emacs &`**.
3. To get a list of newsgroups, select *Read Net News* from the *Tools* menu.

To exit, select *Exit from Gnus* from the *Misc* menu. To read a news group or news item, click on the name of the news-group or a news item; then press RETURN. Online help is also available.

CHAPTER 8

Staying Informed About System Activity

If you choose, you can obtain details about the computer's activity. In particular, you can find out how busy the computer is, who is logged in, and what other users are doing. The information you receive depends on the UNIX workstation or host you are logged on to.

Getting Basic System Information

The uptime command displays some information regarding the status of the computer: the time, how long the machine has been up and running, the number of users currently logged on, and the load average. As more users log on to the system and do work, the load average tends to increase. This gives you a rough idea of how busy the system is.

```
$ uptime
10:24am  up 4 days, 4 min  4 users, load average: 0.00, 0.04, 0.00
$
```

The load average numbers are for 1 minute, 5 minutes, and 15 min CPU cycles received. So in the above example, the 1 minute CPU cycles received is 0.00; the 5 minutes CPU cycles received is 0.04; and the 15 minutes cycles received is 0.00.

Finding Out About Other Users

The **who** command displays who is currently logged in to the computer. From left to right, the first entry is the user's login name, followed by that person's terminal (tty), and the date, time, and location of the login.

```
$who
smith ttyq1 Oct 08 10:33
```

```
jones ttyq2 Oct 08 08:22
root  ttyq3 Oct 08 09:51
$
```

The command **whoami** displays only the line of information for the user who is logged on to that terminal. This is useful in finding who is logged on to an unattended terminal.

```
$ whoami
smith
$
```

The **w** command lists all the users on the system and what they are doing. The first line is essentially the uptime command output. For more information on what the fields in the user lines are, type **man w** at the system prompt.

```
$ w
10:27am & 6lup 4 days, 4 mins, 4 users, load average: 0.00, 0.04, 0.00
User  tty   login@  idle JCPU PCPU what
smith ttyq1 10:23am   47   2:15  w
jones ttyq2 11:45am       3:35  csh -v view
$
```

To find out more information about particular users, use the **finger** command. When you use the command **finger** with no arguments, you get abbreviated information (full name, tty, idle time, login time, office) for each user on the system. If you use the command **finger** with a username as an argument, that user's `.project` and `.plan` files also display. (See section 3.)

```
$ finger
Login Name        TTY Idle When
smith John Smith   q1       Mon 10:23
jones Smith Jones  q2  4    Mon 09:51
root  Super User   q3  51   Mon 09:04
$ finger smith
Login name:  smith          In real life:  John Smith
Office:  Bldg 4 D560 x2345   Shell:  /bin/csh
On since Oct 8 10:23:50 on ttyq1 from 134.3.130.197
58 seconds Idle Time
Plan:
:
<displays user .plan file>
```

```
$
```

A text editor can be used to add messages or information to the `.project` and `.plan` files in your home directory. (For details, see section 5.) These files should only contain information that you would like other users to know about you. Also, be sure that you give permission for others to read your `.project` and `.plan` files. If you don't, people who **finger** your username will not be able to see the information you include in your files. (See section 3.7.2 for more information.)

You can find historical information up to and including yesterday about performance and usage of RS6000/AIX computers (e.g., the Batch Farm, general interactive servers, NFS and AFS servers, tape servers, etc.) that are maintained by SCS at the Web URL *http://www.slac.stanford.edu/grp/scs/slaconly/usage.stats/@Quick-Index-to-Reports.html*. Please note these are accessible only to SLAC nodes.

APPENDIX A

UNIX Command Summary

Beginning and Ending Work Session

login	Logs you in to the computer you want to use.
logout	Logs you off the computer you have been using.

Browsing

head test	Displays first 10 lines of the file named test.
head -25 test	Displays first 25 lines of the file named test.
more longfile	Displays file longfile a screenful at a time; press the space bar to continue
tail test	Displays last 10 lines of the file named test.
tail -5 test	Displays last 5 lines of the file named test.

Command History

history	Reviews most recent commands.
script	Starts recording of any subsequent interaction; type exit to stop recording. The file typescript will hold text of recorded session.

Communication

biff n	Tells the computer not to notify you as soon as you receive any new mail.	
biff y	Tells the computer to notify you as soon as you receive any new mail.	
mail	Enters mail program to read mail.	
	h	shows headers of your mail messages.
	?	shows other options.

	number (the message number)	displays mail message with that number.
	d number (the message number)	deletes mail message with that number.
	s number filename	saves the message with the number you specify in the file filename.
	q	quits the mail program, appending any messages you have read to the mbox file (your personal Mailbox File).
	x	quits the mail program, leaving any messages in the computer Mailbox File (where all mail is stored until it is read).
mail jane	.Sends mail to the user jane on the same computer. ype message body and end with CTRL-d.	
mesg n	Prevents others from using the write or talk command to communicate with you	
mesg y	Allows others to use the write or talk command to communicate with you.	
talk smith	Initiates interactive text discussion with logged-in user jsmith. Use CTRL-c to quit.	
telnet sscvx1	Allows you to connect to the computer SSCVX1.	
write jsmith	Sends short message to logged-in user jsmith. Type messages and end with CTRL-d	

Comparing and Searching Files

cmp file1 file2	Shows the location in file1 where the first difference between the two files occurs.
diff file1 file2	Displays differences between files file1 and file2.
grep computer intro.doc	Displays all lines in file intro.doc that contain the string computer.

Directory Commands

cd ..	Changes to directory one level above.
cd docs/trg	Changes to subdirectory docs/trg.
cd /usr/bin	Changes to /usr/bin directory.
mkdir testdir	Creates a new subdirectory called testdir.
pwd	Prints name of the working directory.
rmdir testdir	Removes subdirectory testdir (must be empty).

File Commands

cat part1.doc part2.doc	Strings together the files part1.doc and part2.doc, writing its output on stdout.
cp srcfile destfile	Copies the file srcfile to destfile.
ftp slacvx	Allows transfer of files to or from remote computer slacvx.
	get ftp command Transfers a file from the remote computer to your computer.

	send ftp command Transfers a file from your computer to the remote computer.
ls	Lists files in directory showing which ones are subdirectories, executables, etc. This is the SLAC alias for ls -F
ls -a	Lists files in directory including dot files.
ls -l	Lists files in directory in long form (i.e., with additional information).
mv oldfile newfile	Moves or renames file oldfile to newfile.
rm badfile.tmp	Permanently removes file badfile.tmp.
rm -i *.c	Removes all files with suffix .c, but asks for confirmation on each file.
sort filename	Reorders the lines contained in the file filename alphanumerically.
spell filename	Spellchecks the contents of the file filename.

Help

aid unix	Pointer to information about UNIX.
apropos files	Same as man -k.
man diff	Displays manual page for the diff command.
man -k files	Displays commands related to keyword files.
man man	Displays information about the online manual.
man csh	Displays documentation on commands that are built into the C shell.
man command	Displays documentation on the specified command.

Miscellaneous

date	Displays the date and time.

Other Users

finger jsmith	Displays user information on jsmith.
finger sysop@slacvx	Displays user information on sysop on remote computer slacvx.
w	Shows who and what all logged-in users are doing.
who	Tells you who is logged in to the computer.
whoami	Displays information on the user logged on to this terminal only.

Password

passwd	Changes your current password.

Print Commands

lpq	Shows default printer queue.
lpq -Pcadsun2	Shows the cadsun2 printer queue.

lpr filename	Prints file filename to the default printer.
lpr -Pcadsun2 filename	Prints file filename to the printer cadsun2.
lprm 34	Removes current user's print job 34 from the default print queue.
ls | lpr -Pcadsun2	Prints the output of the ls command on cadsun2.
pr unix.doc | lpr	Produces headers printed on each page of the file unix.doc.

Security

chmod a+x test	Make file text be executable by all users.
chmod g-r,o-r test	Denies the users in the group and others read access to the file named test.
groups	Displays the group(s) to which you belong.

Shell Related

alias	Shows all current aliases.
alias dir ls -l	Assigns the alias dir to the command ls -l.
unalias dir	Removes the alias for dir.
echo $workdir	Shows the value assigned to shell variable workdir.
exit	Leaves any shell.
history	Shows n most recent commands as specified by value n of history shell variable.
set	With no argument, lists shell variables and their values.
set history=20	Sets the shell variable history to 20.
set noclobber	When added to .cshrc file, prevents accidentally overwriting an existing file with the > character.
set path=(path)	Sets your search path to the directories listed within parentheses. Do not separate directories with commas.
set prompt="prompt"	Sets the prompt to be what you specify within the quotation marks.
set term=vt100	Sets your terminal type to vt100. Use appropriate terminal type.
source .cshrc	Runs the commands in the .cshrc file.
printenv	Prints environment variables and their values.
setenv PRINTER cadsun2	Sets the environment variable PRINTER to cadsun2. Notice the capitalization and spaces.
unsetenv PRINTER	Undefines the environment variable PRINTER; lpr will then use the system default printer.
unalias dir	Remove the alias for dir.

System Activity

uptime	Displays information concerning the status of the computer.

Text Editing

emacs `unix.doc`	Starts the emacs text editor and opens the file `unix.doc` so you can edit it.	
	CTRL-x CTRL-c	Exits the editor; returns you to the prompt.
	CTRL-x CTRL-s	Saves your work. Writes the contents of the buffer (or work space) over the original file.
	CTRL-x CTRL-w	Editor prompts you for the name of the file where it should save the buffer.
	CTRL-z	Suspends the editor. The editor remembers your files and buffers so you can begin where you stopped when you issue the command fg.
fg	Resumes the editor stopped with CTRL-z. The editor remembers your files and buffers so you can resumewhere you suspended.	
vi `unix.doc`	Starts the Vi text editor and opens the file `unix.doc` so you can edit it.	
	:q	Quits the editor if the file hasn't changed.
	:w	Saves your work. Writes the contents of the buffer (or work space) over the original file.
	CTRL-z	Suspends the editor. The editor remembers your files and buffers so you can begin where you stopped when you issue the command fg.
fg	Resumes the editor stopped with CTRL-z. The editor remembers your files and buffers so you can resumewhere you suspended.	
xee `unix.doc` or xeg `unix.doc`	Starts the Uni-Xedit text editor and opens the file `unix.doc` so you can edit it. Starts the Uni-Xedit text editor on an X-Windows system.	
	quit	Quits the editor if the file hasn't changed.
	qquit	Quits the editor without saving changes.
	file	Saves your work.

APPENDIX B *Other Useful Documents*

The following is a summary of documents that were referenced in this UNIX guide. How to print documents is described in "Printing" on page 43.

Title	Availability
Web URL http://www/slac.stanford.edu/comp/unix/unix.html	UNIX Web page: policies, procedures, pointer to other documents, account form, public machines, etc.
SLAC AFS Users' Guide, Rev. 5	/usr/local/doc/intro/afs/afs.ps; available in hardcopy at SCS Help Desk and as HTML doc at URL http://www/slac.stanford.edu/comp/unix/afs/users-guide.html
Documentation about UNIX CPU Farm	Web page URL http://www.slac.stanford.edu/comp/farm/farm.html
AIXwindows and AIXwindows Desktop User's Guide (version 3.2); also try AIX command **`info`**	Computer Building Lobby, SLAC Library
guide to choosing good passwords	/usr/local/doc/policies/password
Using WDSF to Restore Backed Up Files for Unix Users (Including NeXT)	/user/local/doc/how-to-use/WDSF-Restore.ps; also accessible from Web SLAC UNIX page (see above)
GNU Emacs Manual	Computer Building Lobby; softcover manual is available at bookstores
online, interactive vi tutorial	/usr/local/doc/vitutor
SLAC-wide E-mail Routing	/usr/local/doc/how-to-use/Email-Routing
Understanding E-Mail Addresses	/usr/local/doc/intro/Email-Addresses

Index

T

U

V

W

Y