UNIX User's Guide

University Computing & Communications Services
August 1999

Table of Contents

Introduction

Researchers at Bell Labs developed the UNIX operating system in the late 1960's. Originally, Bell Labs released UNIX only to colleges and universities with a complete set of source code and development tools. Because of this, several different versions of UNIX were created such as BSD, SunOS, and System V. We use System V at UCCS.

The purpose of this guide is to:

- Introduce the beginning user to the UNIX system,
- Present an overview of the UNIX file system,
- Provide an overview of the shell commands.

How to Establish an Account

To get started, you first need to establish an account number (login ID) and password. If you are a student, your instructor will provide you with this information. For all other users, you need to apply for an account at the User Services Counter (SQU 322) and fill out a REQUEST FOR USER ACCOUNT form.

Once the account has been established, it may be accessed from any computer on campus or from any remote location by modem. A list of University Computing Labs and lab hours can be found at http://www.csus.edu/uccs/labs/labhours/labhours.htm. Information regarding remote dial in instructions are located at http://www.csus.edu/uccs/saclink.

Information and Assistance

Online documentation is available through the command, **man** + command. An example would be **man ls** (list). The documentation provides a detailed list of specific commands. Press the space key at the end of every page to continue reading. You can use the scroll bar on the right side of the screen to view information that was on previous screens.

Logging On

Once you have established your UNIX account you are ready to log on.

Use a telnet application to access the system. The IP address is ccshp.csus.edu.

Make sure you are using lower case letters, as UNIX is case sensitive.

At the **login** prompt:
    Type your login name and press ENTER

The system will then prompt with **Password**:
    Type your password and press ENTER

If for any reason, they are not considered valid by the system, you will get the error message, **Login incorrect**. The system will prompt you to retry.

> Note: The first time you use your account will be requested to change your password from the assigned Password, to one you choose yourself. After typing your login name and password you will receive a message saying your password has expired. The system will prompt with old password. Type your assigned password once again and follow the on-screen instruction for changing your password.

Logging Off

To log off, type **logout**

Changing Your Password

You can change your password whenever you are logged in. At the system prompt:
    Type **passwd**

The system will respond with, **Old password**:
    Type your current password

You will be told the last time that a new password was successfully entered and be asked to choose a system-generated password or to pick your own. If you pick your own, it must be at least 6 characters with one special character (number or symbol).
The system will respond with, **New password**:
    Type your new password

You will now be prompted with, **Re-enter new password**:
    Type your new password again

If You Lose Your Password

Students who are assigned login ID's and passwords for a class, need to contact their instructor if they need assistance.

Other users will need to file a request for a new password at the User Services Counter, Sequoia Hall, Room 322. Allow three (3) working days for this request to be completed.

File System Structure

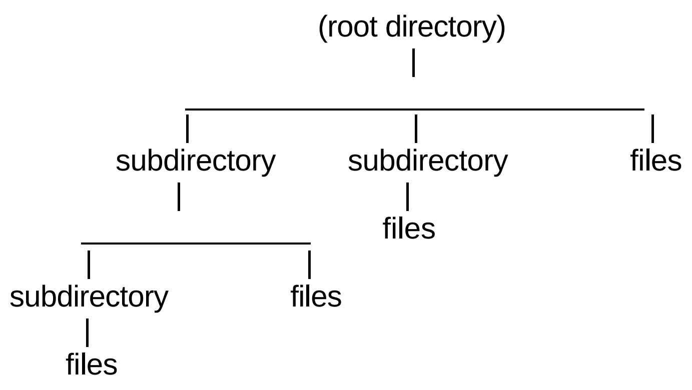
The file system is comprised of a root directory that contains subdirectories and files. These components provide a way to organize, retrieve, and manage information electronically.

A root directory is the main directory. All other directories are under the root directory.

A directory is a group of files, or a work area. Directories can also contain other directories (subdirectories). A directory contains the names of these files and directories. You can make directories, move files into them, rename and delete them.

A file is information stored together with a name. Such as letters, spreadsheets, programs, or data in any format.

The following is an example of what a file system structure might look like:



Login (home) Directory

After you've successfully logged into UNIX, you will be placed at a specific point called the login or home directory. Every user has a unique login directory.

Working Directory

The login directory is not always the working directory. The directory you are working in is your working directory. When you move to a new directory, it becomes your new working directory.

Pathnames

Pathnames are step by step instructions UNIX follows to get from the root directory to the file. An example of this would be: /users/Spillerkj/

An absolute pathname (full pathname) indicates how a file is found beginning from the root directory.

A relative pathname traces the file from the working directory (i.e. from where you are now).

Moving between Directories

First, it might be helpful to know where you are. To find this out, type **pwd**. This will display your working directory.

To get back to your home directory, type **cd**

To change to another directory type, **cd +** directory name

Example: **cd Engineering**   Places you in the Engineering directory

Creating Directories

Before you create a directory, first decide where you want to put it. Use **cd** to get into the appropriate directory you want it listed under and type, **mkdir** + the directory name.

Example: **mkdir docs**   Creates a directory named docs

Deleting Directories

UNIX will not remove a directory if it contains any files and/or subdirectories. Make sure you remove all files and/or subdirectories before you begin**. Be careful, in most cases, you will not be able to recover what you've removed.**

To remove a directory, type **rmdir** + the name of the directory.

Example: **rmdir docs**   Removes the directory named docs

To be safe, use the **–i** option to confirm deletion of the directory, before it's removed.

Example: **rmdir –i docs**   Removes the directory docs with the prompt **docs: ? (Y/N)**

You can delete all of the files and subdirectories in a file using the option **–r** (recursive).

Example: **rmdir –ir docs**	Removes all of the files and subdirectories of the directory docs with the following prompts:		
	directory prog: ? (y/n)	Asks if you are pointing at the directory prog	
	prog/rep.txt: ? (y/n)	Removes file rep.txt	
	prog: ? (y/n)	Removes directory prog	

Listing the Contents of a Directory

Once you get to the directory you are looking for, you can view the names of the files and/or subdirectories under it by typing, **ls**. This will list the names of the files under this directory.

The following are options that can be used with the **ls** command:

Option	Description
-a	List all files, including invisible
-l	List all files and some of their attributes
-s	List all files an their size. Size is in 512-byte blocks
-t	List all files in order of time last modified
-C	List file in multiple columns with entries sorted down the columns.
-F	Places a slash (/) after directory files and an asterisk (*) after executable files.
-R	Include a listing of files in all subdirectories.

You can use more than one option at a time.

Example:	**ls -al**	List all files, including invisible files and their attributes.
	or	
	ls -a -l	List all files, including invisible files and their attributes.

Listing Files to the Screen

The cat (concatanate) command will list a file to the screen. The file is listed continuously from beginning to end.

cat

Example:

cat prog1	Displays the entire file prog1. To pause the display, press CTRL + **s** (press the control key and type s) . To resume the listing, press CTRL + **q**. To abort the listing entirely, press CTRL + **c**.

more

For very long file, use the **more** utility.

Example:

more prog1	Displays one full screen of data from the file, then displays, "—More—(x%)" (Where 'x' represents the percentage of the file viewed so far). Press the space bar if you wish to see the next page of the file or type **q** to terminate the listing. Other options are available when using the more utility. To see what they are, type an **h** at the "—More—(x%)" prompt.

tail

To display the last few lines of a file, use the **tail** command. The + or – preceding a number will start at that line relative to the file.

The options you can use are:

Option	Description
b	(blocks) count by blocks
c	(characters) count by characters
l	(lines) count by lines (this is the default)

Examples:

tail +5 prog1	Displays the file prog1 beginning at line 5.
tail -5 prog1	Displays the last 5 lines of the file prog1.
tail -c -5 prog1	Displays the last 5 characters of the file prog1.

Wildcards

Wildcards are special characters that make typing of multiple file names somewhat easier, they have a special meaning in file names. Wildcards commonly are used with the **ls**, **cp**, **rm** and **mv** commands.

UNIX handles the **?** and ***** wildcards in the same way for every command, unlike DOS where you have to memorize which commands can handle wildcards and which ones can't. UNIX also, recognizes letters after the *****, again unlike DOS. There are two wildcards:

?	means "any single letter."
*****	means "anything at all."

Examples:

ls prog*	Lists all directory entries beginning with "prog" and ending with any character sequence.
ls *prog	Lists all directory entries beginning with any character sequence and ending with "prog".
ls *prog*	Lists all directory entries which contain the character string "prog", preceded and/or followed by any other characters.
ls *	Lists all directory entries and all subdirectory entries. Note: The ***** works the same as *.* used in DOS.

The **?** Pattern matches any single character. Each **?** represents only one character.

Examples:

ls ?	Lists all files that have single-character names, just as **ls ??** would list any two-character names.
ls prog?	Lists all files beginning with "prog" and followed by any single character
ls student??	Would match any file name with student + two digits, such as student98 and student99. It would not list student2 or student395.

Printing

The **prtbanner** command allows you to obtain a hard copy listing of files or programs. To print a file, type: prtbanner + filename.

Example: **prtbanner prog2** The file prog2 will print. You will be prompted with the following:

This routine allows you to place your **name or BIN number** on the banner page of your printout. If you leave the fields blank, the only way to identify your printout will be by the userid on the banner page.

Name can NOT have any spaces in it. If you want to put first and last name in they must run together – JohnBrown - or be connected – John_Brown. Spaces will cause an error.

You will probably want to experiment to see how the banner looks with both the name and BIN number. If you put too many characters in they will get truncated.

Enter your name, or leave blank. Enter your 2 digit BIN number, or leave blank. Enter pathname of file name to be printed

Copying Files

You can take the contents of one file and make a duplicate copy to another file. To copy a file from another directory, the pathname needs to be included with the exception of your current working directory. You can eliminated the path name you are copying to, if it is in your working directory by typing a period (.) at the end of the command line.

Examples:

cp prog1 prog2	Makes a new copy of prog1 and names it prog2.
cp /docs/prog1.txt	Copies the contents of the file prog1 from the docs directory into the current working directory.
cp prog1 test/prog2	Copies the file prog1 into the test directory and names it prog2.

To append one file to the end of another:

cat prog1 prog2 > prog3	Creates a new file call prog3 that contains the contents of prog1and prog2.

Renaming and Moving Files

You can rename a file in the same directory or move a file from one directory to another.

To rename a file, type **mv** + the filename + new filename.

Example: **mv prog1 prog2**	Changes the name of file prog1 to prog2	***Beware: Any existing files named prog2 will be overwritten***

Deleting Files

You can remove an unwanted file by using the **rm** command. **Use caution when you delete; once a file is gone, there maybe no way to get it back.**

To delete a file, type **rm** + filename.

Example: **rm prog1**	Removes the file prog1

To be safe, use the **–i** option to confirm deletion of the file, before it's removed.

Example: **rm –i prog1**	Removes the file prog1 with the prompt **prog1: ? (Y/N)**

Suspending and Aborting I/O Process

UNIX provides a way to control running processes. Hold down the control CTRL key and press a character key.

CTRL + **c**	Aborts the process currently running at your terminal
CTRL+ **s**	Temporarily stops output form printing on the screen
CTRL + **q**	Resumes printing after typing CTRL + **s**
CTRL + **z**	Suspends the current job. Your Job may be resumed by typing the command **fg**

File Access Permissions

UNIX was designed from the beginning to be used by more than one person and has the ability to designate permissions for file access. Permissions can be used to determine who can use which file or directory and how they can use it.

- Read permission allows you to look at a file or directory.

- Write permission allows you to make changes to a file. Although, you might not be able to delete or rename it. If you have write permission in a directory, you can create new files and/or delete from it.

- Execute permission lets you run the program contained in the file. The program can be a real program or a shell script. If the file doesn't contain a program, execute permission doesn't do anything. For a directory, execute permission lets you open files in the directory and use **cd** to get to the directory to make it your working directory.

Every file and directory has an owner. The owner is usually the person who made the file or directory. You can change who owns a file in the next section, Protecting Files.

Every UNIX user is a member of a group. To see which group you are in, type **id**.

To see who has what permission to a file, type **ls –l** + the filename.

Example:	**ls –l prog1**	Lists permission of the file prog1
	ls –l	List permission for the entire directory contents of the current working directory

The system will show you the following:

```
-rw-r--r--        1 Moorelj  user 3152   Aug 17   11:12    prog1
```

In any digit:

A hyphen **(-)** indicates no permission
An **r** indicates read permission
A **w** indicates write permission
A **x** indicates execute permission

Digit 1	Is a hyphen (-) if it is a file, and a d if it is a directory.
Digit 2	Shows **owner** read permission
Digit 3	Shows **owner** write permission
Digit 4	Shows **owner** execute permission
Digit 5	Shows **group** read permission
Digit 6	Shows **group** write permission
Digit 7	Shows **group** execute permission
Digit 8	Shows **other** read permission
Digit 9	Shows **other** write permission
Digit 10	Shows **other** execute permission

Protecting Files

You can change the permissions of a file you own using the chmod (change mode) command. You can enable or disable read, write and execute permissions for owners, groups, and others.

The following are options that can be used with **chmod** command:

Options	Description
u	Owner of the file
g	Group to which the owner belongs
o	Other users
a	(all) Can be used in place of u, g, and o
+	Add permission for specified user
-	Remove permissions for specified user
=	Set permission for the specified user – all other permissions for that user are reset.
r	Read permission
w	Write permission
x	Execute permission

Examples:

chmod ugo+r prog1	Indicates that the user/owner, group, and others can read prog1
chmod go-w prog1	Prevents anyone except the user/owner from changing prog1
chmod u+rwx prog1	Gives the user/owner, read, write and execute permission for prog1
chmod o-wx prog1	Removes others from write and execute permission for prog1. If they had read permission before this command was given, they will still have read permission
chmod go-x prog1	Removes groups and others execute permission for prog1

Sorting and Merging Files

The **sort** command sorts a file line-by-line in alphabetical order. It alphabetizes all the lines according to the first letter or letters in each line. Each line in the file is unaffected, only the order of the lines changes. The sort command can also merge files and then sort them.

The following are options that can be used with the **sort** command:

Options	Description
-c	(check) Check only to see if the file is sorted. If not, a message is returned
-f	(fold) Folds all uppercase characters into lowercase and then sorts
-n	(numeric) Numeric sort for data, "+","-" are assigned their arithmetic meaning
-tx	(set delimeter character) Changes the field delimiter from the default (a blank) to the character specified by x
-r	(reverse) Reverses alphabetical order
-o	(output) Indicates where to put the sorted output

Merge example:

sort fall97 fall98 fall99	Merges the files, fall97, fall98 and fall99 and then sorts them.

Sort examples:

sort fall99	Sorts the file fall99 in ascending order.
sort fall99.stu	Sorts the file fall99.stu

From	TO
Stamp, Lynne	Andrews, Dave
Wallace, Susan	Looper, Joey
Moore, Dave	Looper, Laura
Andrews, Dave	Moore, Dave
Spiller, Kevin	Spiller, Kevin
Looper, Laura	Stamp, Lynne
Looper, Joey	Wallace, Susan

Finding Files and Subdirectories

There are two programs that can help you find files: **find** and **grep**.

find

You can use the **find** command to look for files and/or directories. **Find** looks in the directory you specify, and in all that directory's subdirectories.

If you wanted to find the file prog1 and are not sure what subdirectory it's located in you could type:

Examples:
find / -name prog1.txt
Looks in the root directory and all subdirectories for the file prog1.txt

Beware, this could take a very long time. It is better to specify where to look.

find /user/kevin – name prog1.txt
Looks in the root directory user, directory kevin and all of its subdirectories for the file named prog1.txt

find /user/kevin /user/sharon –name prog1.txt
Looks in the root directory user., directory kevin and all of its subdirectories; and in the directory sharon and all of its subdirectories for the file prog1.txt

If you know only part of the file name:

find /user/kev/docs –name "prog*"
Looks in the directories user, kev, docs for all files that begin in prog

To find a lost directories:

find / -name "doc*" – type d
Looks in the root directory and all subdirectories for directories that begin in doc

grep

The **grep** command (global/regular expression and print) looks inside files and searches for a series of characters (i.e. character strings). Every time it finds a line that contains the specified characters, it displays the line on-screen. If it is looking in more than one file, **grep** also tells you the name of the file in which the characters occur. You control which files it looks in and which characters it looks for.

There are three types of grep programs: grep, egrep and fgrep. This document only discusses grep.

The following are options that can be used with the **grep** command:

Option	Description
-c	(count) Displays only a count of how many lines were matched in a file
-e	(expression) Allows the pattern to begin with a hyphen. (Normally only options begin with a hyphen)
-l	(list) Displays the name of each file that has at least one match
-n	(number) Precedes each line by its relative line number in the file
-s	(status) Returns an exit status value without any output. The status values are, zero if a match is found, one if no match is found, and two if an error occurs in accessing the file
-v	(reverse search) Displays only unmatched lines form the files specified
-y	Ignores case differences when searching for a match (i.e. uppercase characters will match with lower case characters)

Examples:

grep Abc prog1.txt prog2.txt

Displays each line in prog1.txt and prog2.txt that contains an uppercase A followed by a lowercase b and c. Each line displayed will be preceded by the name of the file it came from (either prog1.txt or prog2.txt)

grep -l Abc *

Displays only the name of each file that contains the pattern Abc

grep "Abc" *

Looks in the working directory but not all of its subdirectories for the characters Abc

grep –n '153*0' prog1.txt

Displays each line in prog1 that contains a string beginning with 153 and ending with 0. Each line will be preceded by its line number

Note: when using wildcards in the pattern, it is best to enclose the entire pattern in single quotes. Wildcard characters have special meaning to the shell.

Redirecting Input and Output

Unix refers to the user's terminal as the standard input and output device. Unix provides commands that permit you to redirect the standard input from your terminal or output to your terminal.

Redirecting Standard Input

To redirect the output of a command (which is usually on-screen) to a file, use the > character.

Example: **ls > directoryfile** The results of the ls command will be written to the file, directoryfile

Redirecting Standard Input

The redirect input symbol (<) indicates to UNIX that the input of a command or program is to come from a file

Example: **cat < prog1.txt** The input for the cat command will be prog1.txt

Using Redirect Output and Input Together

The redirect output symbol > an the redirect input symbol < can be used together:

Example: **cat<prog1>prog2** The input for the cat command will be prog1 and the output will be placed in prog2

Beware: any prior existing prog2 will be overwritten.

Appending Standard Output to a File

The redirect output symbol > will overwrite any information in the file to which output has been redirected. The append output symbol >> causes the new information to be added to the end of a file, leaving intact the information that was already in the file.

If prog1 and prog2 already exist, the following example will append prog2 to the end of prog1:

Example: **cat prog2>>prog1** prog2 will be the input to the cat command and the output (a listing of prog2) will be appended to prog1

Pipes

The pipe symbol (|) is used to connect the output of a command to the input of another command. This symbol is often located on the keyboard with the key \.

If you type two commands separated by a |, you tell UNIX to use the output of the first command as input for the second command.

Example: **cat prog1.p|grep procedure|sort|more**

The output of the cat command (the pascal program prog1.p) is piped to the grep command, the output of the grep command (a listing of all lines contained the string "procedure") is then piped to the sort command, which will print the sorted listing of all procedure names contained with in prog1.p to the screen.

The redirection symbols can be used along with the pipe symbol as follows:

Example: **cat listowords | sort > sortedwords** The file listowords is piped to the sort command and the sorted output listing is redirected to the file sortedwords.

Displaying Process Status

The **ps** command display the status of all the active processes you control. The following information is displayed for each process:

PID	Process ID number
TTY	Terminal ID number that controls the process
stat	State of process
TIME	The number of seconds the process
COMMAND	The name of the command that started the process

The following are options that can be used with the **ps** command:

Option	Description
-a	(all) Displays status information for all active processes controlled by a terminal
-l	(long) Displays a complete status report
-f	(full) Displays full listing, more than just **ps**

Examples:

ps –a	Displays the state for all active processes attached to the users terminal
ps –u kevin	Displays all processes belonging to user kevin

Interrupt or Terminating Active Processes

If you need to interrupt a process type, CTRL + **c**. Hold down the CTRL key and press c.

The quit command kills the program and a file of the terminated process. Type **quit**, the system will then give you the message, **Quit (core dumped*)***. This message tells you that the process is dead and the ceased body of process has been put in a file named core.

The **kill** command is used to stop active process. First you need to know the name of the process, PID (process ID), type **ps PID**. You will get back something like:

```
PID     TTY     TIME   COMMAND
14398  14       0:09   –csh
14790  14        0:07  cc Prog1
14930  14        0:02  ps
```

Then to stop the process, type **kill** + the PID

Example: **kill 14790** Kills the PID 14790

Miscellaneous

Command	Description
date	Displays the current date and time
hostname	Displays the name of the UNIX system on which your login resides.
pwd	To find out where you are
printenv	Displays the current working environment variables.
who	Displays the login names of the users logged in to the UNIX system, along with the terminals being used and login times.
whoami	To find out who you are logged in as.
wc	Displays the size of a file in lines, words, or characters.

Options	Description
-c	(characters) Displays the number of characters in a file.
-l	(lines) Displays the number of lines in a file.
-w	(words) Displays the number of words in a file.

Example: **wc –l prog2** Displays the number of lines in prog2.
The system responds back with "***18 prog2***".
There are 18 lines on prog2.

Appendix A
SUMMARY OF UNIX COMMANDS

cat	Displays contents of a file
cd	Changes working Directory
chmod	Changes file access permissions
cp	Copy a file to a new file
CTRL+ **c**	Abort current job
CTRL+ **q**	Resume suspended input to the terminal
CTRL+ **s**	Suspends input to the terminal
CTRL+ **z**	Suspends the current job
date	Display current date and time.
find	Looks for the files you specify.
grep	Search for a pattern
help	Provides a brief description of system commands
hostname	Displays the user's login system
kill	Abort a process
logout	Log out of the current shell
ls	List directory
man	Provides detailed descriptions of a given command
mkdir	Create a directory
more	Displays one page at a time
mv	Rename or move a file
passwd	Change password
printenv	Displays current environment
prtbanner	Print a file on the line printer
ps	Displays process status
pwd	Displays current environment
rm	Delete a file
rmdir	Delete a directory
setenv	Set an environment variable
sort	Sorts a file
tail	List the end of a file
wc	Display the size of a file in three different formats (lines, words, and/or characters)
who	Displays the current system users.

Appendix B
COMMAND LINE OPERATORS

Command	Description
*****	Wild card
?	Wild card
>	Redirect command output to a file
<	Redirect input to a command
>>	Append to end of file
|	Pipe command output to input of another command