# Basic Unix Commands

## Contents

## 1 Intro

Unix is a multi-tasking, multi-user operating system. This means that on any given computer, it appears as if many things are happening at once and that there may be more than one person logged into the computer at once. Regardless of which machine you log into in the Institute, you will have access to your user files and the basic commands will be identical.

The Unix file system is hierarchical. Every file is stored in a directory. A directory can contain many files or none at all, and may also contain other directories called subdirectories. Unix has three types of files:

**Normal Files** These are data files which might contain text, source code, executable files, *etc*.

**Special Files** These represent physical devices such as terminals and disk drives. The “device drivers” will translate any references to such files into the hardware instructions needed to carry out the tasks.

**Directories** These contain “pointers” to normal files, special files and other directories.

File names can be as long as you like, unlike in MS-DOS.

The parent directory of all other directories is called the “root” directory and is denoted by /. Every file on the system can be accessed by tracing a path from this root directory. See the section on pathnames.

Each user has a “home” directory: this is where you will be located when you login. You are then free to traverse the directory structure of the Institute and to add to and change the part of the structure which you own.

Unix commands have the following format:

command [*options*...] [*arguments* ]

Where command is the command name, *options* refer to optional command modifiers (usually prefixed by a “-”) and *arguments* are the optional or required command parameters (often file names). Spaces or tabs are required between commands and options and between options and each argument. Commands must be typed in the proper case (nearly always lower case).

Although each option to a command may be prefixed by a “-”, with almost all of the standard commands you may put the options together. For example

```
command -a -b -c -d
```

can often be abbreviated as

```
command -abcd
```

Again, this is almost always the case - the most notable exceptions are in the GNU commands enscript and gcc.

The following are a series of Unix commands which will help you use the computers. They are given in their most basic form and more information will be available from their on-line manual pages (accessed through the man command described below).

Each command will be given in a generic form, perhaps with an example of an actual usage. In the examples, the line alph% will indicate the prompt you see on your screen (which you would not type in if actually using the command). Note that your prompt may be different.

## 2 man - Accessing On-Line Manual Pages

The man command looks up the manual page for a command.

The format of man is

man [-k] *name*...

**-k** Prints a list of all manual pages containing the keyword *name*.

Use the *-k* option if you do not know the name of the command or program.

## 3 pwd - Print the Working Directory

The pwd command prints the full pathname of your current working directory. The format of pwd is

```
pwd
```

For example,

```
alph%  pwd
/home/stlawrence/user/newuser
alph%
```

The meaning of a full pathname is described in the section on "Filenames and Pathnames".

## 4 cd - Changing Directory

The cd command changes the current working directory to the directory specified. The format of cd is

cd [*directory*]

If you do not specify *directory*, cd changes to your home directory.

For example,

```
alph% cd /home/stlawrence/user/newuser
alph% pwd
/home/stlawrence/user/newuser
alph% cd Mail
alph%  pwd
/home/stlawrence/user/newuser/Mail
alph% cd
alph%  pwd
/home/stlawrence/user/newuser
alph%
```

## 5 ls - Listing the Contents of Directories

The ls command lists the contents of one or more specified directories or lists information about one or more specified files. Normally ls does not list filenames that begin with a dot ("."). The format of ls is

ls [-*aFl*] [*df(1)*] [*df(2)*] [...] [*df(n)*]

where *df(1..n)* is a directory name or a file name.

**-a** Lists all files, including those that begin with a ".".

**-F** Marks directories with a / and executable files with a *.

**-l** Produces a longer, more informative listing (see the section on chmod for more information).

For example,

```
alph% ls
Mail a.out year1 zeta.f zeta.o
alph% ls -F
Mail/ a.out* year1 zeta.f zeta.o
alph%
```

## 6 mkdir - Making Directories

The mkdir command makes directories with specified names. The format of the mkdir command is

mkdir *directory(1) directory(2)... directory(n)*

For example,

```
alph% ls -F
Mail/ prog/ zeta.f
alph% mkdir thesis zeta
alph% ls -F
Mail/ prog/ thesis/ zeta/ zeta.f
alph%
```

## 7 rmdir - Removing Directories

The rmdir command removes *empty* directories with specified names. The format of the rmdir command is

rmdir *directory(1) directory(2)... directory(n)*

The rmdir command will not remove a directory with files in it - for this use the rm -r command, described later in this section, but **be careful!**.

For example,

```
alph% ls -F
Mail/ prog/ thesis/ zeta/ zeta.f
alph% rmdir zeta
alph% ls -F
Mail/ prog/ thesis/ zeta.f
alph%
```

## 8 cp - Copying a File

The cp command makes a copy of a file or copies multiple files into a directory. The format of the cp command is

cp *source-file destination-file*

or

cp *source-file(1) source-file(2)... source-file(n) destination-directory*

The first form makes a copy of the file *source-file* called *destination-file* and the second copies series of files *source-file(1..n)* into directory *destination-directory*.

```
alph% ls -F
alpha.f beta.c mydir/ zeta.f
alph% cp zeta.f zeta.f.old
alph% ls -F
alpha.f beta.c mydir/ zeta.f zeta.f.old
alph% cp alpha.f zeta.f mydir
alph% ls -F mydir
```

```
alpha.f zeta.f
alph%
```

## 9 cat - Printing Files Onto the Screen

The cat command prints out the contents of a series of files one after the other. The format of the cat command is

cat *filename(1) filename(2)... filename(n)*

For example, to read the "message of the day",

```
alph% cat /etc/motd
```

## 10 more - Printing Files One Screen at a Time

The more command prints out the contents of named files, one screen full at a time. The format of the more command is

more *filename(1) filename(2)... filename(n)*

To quit more, press the **q** key, to move one line at a time press the **RETURN** key or the **SPACE** bar to move one screen full at a time.

## 11 mv - Moving and Renaming Files

The mv command changes the name or location of a file or directory. The formats of the mv command are

mv *oldfile newfile*

mv *file(1) file(2)... file(n) directory*

mv *olddir newdir*

For example,

```
alph% ls -F
Mail/ prog/ zeta/ zeta.f zeta.f.old
alph% mv zeta.f.old zeta-old.f
alph% ls -F
Mail/ prog/ zeta/ zeta.f zeta-old.f
alph% mv zeta.* zeta
alph% ls -F
Mail/ prog/ zeta/
alph% ls -F zeta
zeta:
zeta.f zeta-old.f
alph% mv prog program
alph% ls -F
Mail/ program/ zeta/
alph% mv zeta program
alph% ls -F
Mail/ program/
alph% ls -F program
program:
zeta/
alph%
```

## 12 rm - Removing Files and Directories

The rm command removes files and directories. **Caution:** There is no way to reverse this process (although see the section on backup). The format of the rm command is

rm [-i] [-r] *fd(1) fd(2)... fd(n)*

where fd(1..n) are files or directories.

**-i** Inquire before removing a file ("y" to delete, anything else to not delete).

**-r** Recursively remove a directory and all its contents and subdirectories (**Use with extreme care**).

For example,

```
alph% ls
Mail/ prog.f
alph% rm prog.f
alph% ls
Mail/
alph%
```

## 13 chmod - Changing Access Permissions

The chmod command changes the "permissions" on a file or directory. It gives or removes access for another user or group of users to read, change or run one of the files owned by you.

Users on the system fall into three categories:

**user** You.

**group** Anyone in the same class as yourself, such as *pg_1999*, *staff* or *postdoc*.

**other** Anyone who uses the Institute Computers (sometimes called **world**).

The format of the chmod command is

chmod *ugo+-=rwx fd(1) fd(2)... fd(n)*

where fd(1..n) may be a file or directory.

where

**ugo** Specify **u** (user), **g** (group) or **o** (other).

**+-=** Specify **+** (add), **-** (subtract) or **=** (set).

**rwx** Specify **r** (read), **w** (write) or **x** (execute).

For files, read permission means you can read the contents of a file, write permission means you can change the file and execute permission means you can execute the file (if it is executable or it is a shell script).

For directories, read permission means you can see what files are in the directory, write permission means that you can add to and remove files from the directory and execute means you can access files in that directory.

Note that to access a file you must have execute permission on all the directories above that on the file system (including the one in which it is resident). In addition, you must have the appropriate access permissions for that file.

The permissions on a file may be shown by using the command

```
alph% ls -lgF
...
-rw-r--r-- 1 newuser pg 1999 1451 Jan 18 11:02 phone
drwxr-xr-x 2 newuser pg_1999 512 Jan 22 12:37 thesis/
...
alph%
```

There are 10 fields at the start of the entry. The first letter refers to whether the entry is a file (-), directory (d), or something else (such as s or l). The next letters should be considered in groups of three.

The first group of three refers to the user. The second group of three refers to the members of the group "pg_1999". The last group of three refers to world.

Inside each group of three, the three entrys refer to read (r), write (w) and execute (x). A hyphen (-) indicates something not being set. In the example above all users on the system can read and change into the directory called *thesis* and all users can read the file *phone*, and in addition the owner can write in both the *thesis* directory and can change the file *phone*.

If, for example, the material in the file *phone* was personal, it is possible to make sure no one else can read the file by typing

```
alph% chmod g-r o-r phone
```

and the new output of ls -agl would be

```
alph% ls -agl
...
-rw------- 1 newuser pg_1999 1451 Jan 18 11:02 phone
drwxr-xr-x 2 newuser pg_1999 512 Jan 22 12:37 thesis
...
alph%
```

in this case g-r refers to "group remove read" and o-r "others remove read". You could equally use

```
alph% chmod g+r
```

to allow anyone in the group "pg_1999" to read the file.

## 14 diff - Finding the Differences Between Two Files

The diff command compares the contents of two files. The format of diff is

diff *file1 file2*

For example,

```
alph% cat dataA
My travel plans are as follows:
Oxford - Heathrow by bus.
Heathrow - Paris by plane.
alph% cat dataB
Travel Plans:
Oxford - Heathrow by bus.
Heathrow - Paris by plane.
alph% diff dataA dataB
1c1
< My travel plans are as follows:
---
> Travel Plans:
```

The diff command compares files on a line-by-line basis. A < precedes lines from *file1* and a > precedes lines from *file2*.

## 15 grep - Searching for Strings in Files

The grep command scans a file for the occurrence of a word or string and prints out any line in which it is found. The format of grep is

grep [-i] '*string*' *filename(1) filename(2)... filename(n)*

**-i** Ignore case in the search

The single right-quotes around *string* are necessary only if the string contains non-alphanumeric characters (such as **SPACE**, **&**, *etc.*).

For example,

```
alph% cat dataA
My travel plans are as follows:
Oxford - Heathrow by bus.
Heathrow - Paris by plane.
alph% grep us dataA
Oxford - Heathrow by bus.
alph% grep 'are as' dataA
```

```
My travel plans are as follows:
alph%
```

## 16 wc - Counting Words

The wc command counts the lines, words and characters in a file. The format of wc is

wc [-l] [-w] [-c] *filename*...

**-l** Prints the number of lines in the files.

**-w** Prints the number of words in the files.

**-c** Prints the number of characters in the files.

If no options are specified, wc prints out all three.

For example,

```
alph% wc dataA
3 16 85 dataA
alph% wc –l dataA
3 dataA
alph%
```

## 17 Shells

Once you have understood the level of windows you might then consider exactly how you interface with the computer to execute the basic Unix commands. This is achieved by something called a shell. A shell is a program which can run within an xterm. It is not only a command interpreter but can also be used as a programming language. Inside the shell you type the basic Unix commands listed above. However, many shells (and there are many of them) have additional features such as automatically finishing the typing of words, the setting of “aliases” which abbreviate commonly used commands and the ability to keep a history of commands that you type so that the commands may be used again without retyping them.

There are four shells currently available in the Institute:

- sh. This is the basic Unix shell (the “Bourne” shell) with very few features.
- csh. This is the “C-shell” with extensions based on the C programming language.
- tcsh. This is the “Extended C-Shell” which contains many improvements on csh. This is the default shell in the Institute which uses the file .cshrc for its configuration.
- bash. This is a hybrid between tcsh and sh which is preferred by some people - the “Bourne Again Shell”.

More information can be found on these by looking up their appropriate manual pages.