Contents

UNIX One

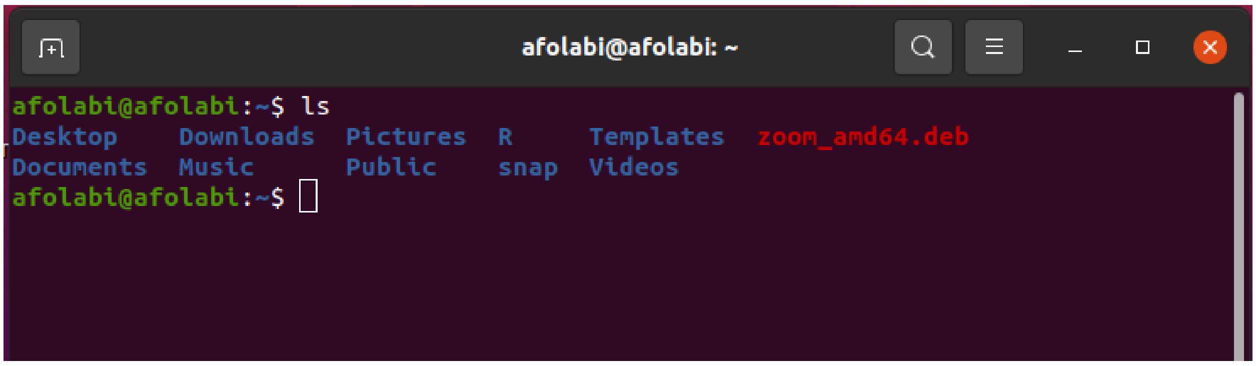
1.1 Listing files and directories

ls (list)

When you first login, your current working directory is your home directory. Your home directory is your username.

To find out what is in your home directory, type in the command prompt (terminal)

```
$ ls
```



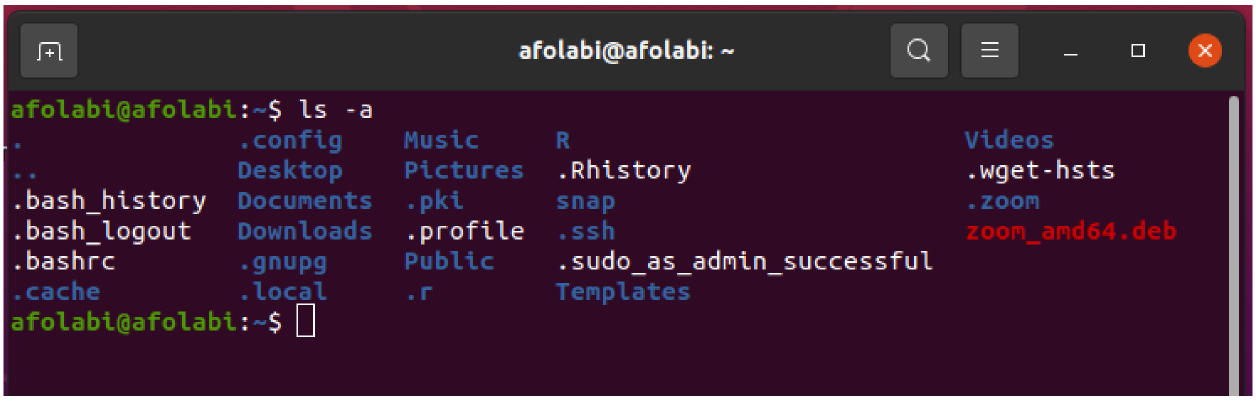
```
afolabi@afolabi:~$ ls
Desktop    Downloads  Pictures  R     Templates  zoom_amd64.deb
Documents  Music      Public    snap  Videos
afolabi@afolabi:~$
```

The **ls** command lists the contents of your current working directory.

ls does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as **hidden files** and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with UNIX!!!

To list all files in your home directory including those whose names begin with a dot, type

```
$ ls -a
```



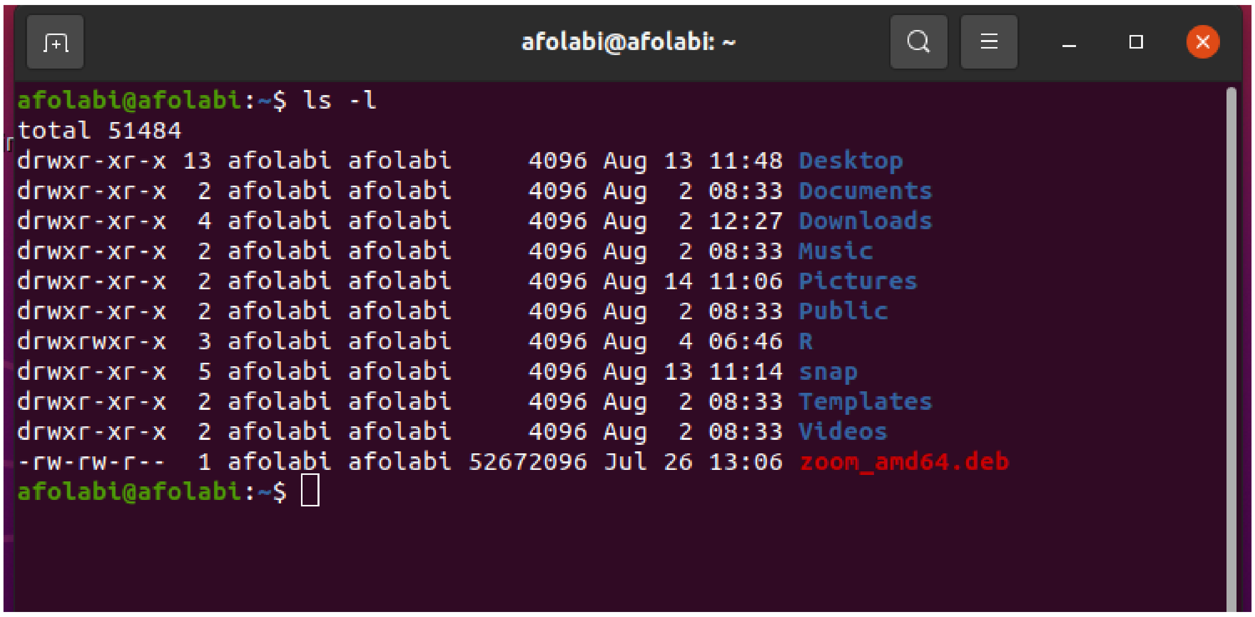
```
afolabi@afolabi:~$ ls -a
.              .config    Music     R                          Videos
..             Desktop    Pictures  .Rhistory                  .wget-hsts
.bash_history  Documents  .pki      snap                       .zoom
.bash_logout   Downloads  .profile  .ssh                       zoom_amd64.deb
.bashrc        .gnupg     Public    .sudo_as_admin_successful
.cache         .local     .r        Templates
afolabi@afolabi:~$
```

As you can see, **ls -a** lists files that are normally hidden.

ls is an example of a command which can take options: **-a** is an example of an option. The options change the behaviour of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behaviour of the command. (See later in this tutorial)

To list all files in your home directory with various properties of each file
Type

```
$ ls -l
```



```
afolabi@afolabi:~$ ls -l
total 51484
drwxr-xr-x 13 afolabi afolabi     4096 Aug 13 11:48 Desktop
drwxr-xr-x  2 afolabi afolabi     4096 Aug  2 08:33 Documents
drwxr-xr-x  4 afolabi afolabi     4096 Aug  2 12:27 Downloads
drwxr-xr-x  2 afolabi afolabi     4096 Aug  2 08:33 Music
drwxr-xr-x  2 afolabi afolabi     4096 Aug 14 11:06 Pictures
drwxr-xr-x  2 afolabi afolabi     4096 Aug  2 08:33 Public
drwxrwxr-x  3 afolabi afolabi     4096 Aug  4 06:46 R
drwxr-xr-x  5 afolabi afolabi     4096 Aug 13 11:14 snap
drwxr-xr-x  2 afolabi afolabi     4096 Aug  2 08:33 Templates
drwxr-xr-x  2 afolabi afolabi     4096 Aug  2 08:33 Videos
-rw-rw-r--  1 afolabi afolabi 52672096 Jul 26 13:06 zoom_amd64.deb
afolabi@afolabi:~$ 
```

Here is the information about all the listed columns:
1. First Column: represents file type and permission given on the file. Below is the description of all type of files.
2. Second Column: represents the number of memory blocks taken by the file or directory.
3. Third Column: represents owner of the file. This is the Unix user who created this file.
4. Fourth Column: represents group of the owner. Every Unix user would have an associated group.
5. Fifth Column: represents file size in bytes.
6. Sixth Column: represents date and time when this file was created or modified last time.
7. Seventh Column: represents file or directory name.

In the ls -l listing example, every file line began with a d, -, or l. These characters indicate the type of file that's listed.

Prefix	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication.

1.2 Making Directories

mkdir (make directory)

We will now make a subdirectory in your home directory to hold the files you will be creating and using in the course of this tutorial. To make a subdirectory called “nimrworkshop” in your current working directory type

```
$ mkdir nimrworkshop
```

To see the directory you have just created, type

```
$ ls
```

1.3 Changing to a different directory

cd (change directory)

The command `cd directory` means change the current working directory to *'directory'*. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type

```
$ cd nimrworkshop
```

Type `ls` to see the contents (which should be empty)

Exercise 1a

Make another directory inside the **nimrworkshop** directory called **biomedicine**

1.4 The directories . and ..

Still in the **nimrworkshop** directory, type

```
$ ls -a
```

As you can see, in the **nimrworkshop** directory (and in all other directories), there are two special directories called (**.**) and (**..**)

The current directory (.)

In UNIX, (**.**) means the current directory, so typing

```
$ cd .
```

NOTE: there is a space between cd and the dot

means stay where you are (the **nimrworkshop** directory).

This may not seem very useful at first, but using (**.**) as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.

The parent directory (..)

(..) means the parent of the current directory, so typing

```
$ cd ..
```

1.5 Pathnames

pwd (print working directory)

Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type `cd` to get back to your home-directory and then type

```
$ pwd
```

The full pathname will look something like this -

```
/home/username
```

Exercise 1b

Use the commands `cd`, `ls` and `pwd` to explore the file system.

1.6 More about home directories and pathnames

Understanding pathnames

First type cd to get back to your home-directory, then type

```
$ ls nimrworkshop
```

to list the contents of your **nimrworkshop** directory.

Now type

```
$ ls biomedicine
```

You will get a message like this -

```
biomedicine: No such file or directory
```

The reason is, **biomedicine** is not in your current working directory. To use a command on a file (or directory) not in the current working directory (the directory you are currently in), you must either `cd` to the correct directory, or specify its full pathname. To list the contents of your **biomedicine** directory, you must type

```
$ ls nimrworkshop/biomedicine
```

~ (your home directory)

Home directories can also be referred to by the tilde **~** character. It can be used to specify paths starting at your home directory. So typing

```
$ ls ~/nimrworkshop
```

will list the contents of your **nimrworkshop** directory, no matter where you currently are in the file system.

What do you think

```
 ls ~
```

would list?
What do you think

```
$ ls ~/..
```

would list?

Summary

Command	Meaning
`ls`	list files and directories
`ls -a`	list all files and directories
`mkdir`	make a directory
`cd` ***`directory`***	change to named directory
`cd`	change to home-directory
`cd ~`	change to home-directory
`cd ..`	change to parent directory
`pwd`	display the path of the current directory

UNIX Two

2.1 Copying Files

cp (copy)

`cp` ***`file1 file2`*** is the command which makes a copy of **file1** in the current working directory and calls it **file2**

What we are going to do now, is to take a file stored in the *tutorial* directory in *examples* directory which is in the *Documents* and use the **`cp`** command to copy it to your **nimrwokshop** directory. First, **`cd`** to your **nimrwokshop** directory.

```
$ cd ~/nimrworkshop
```

We are going to be working with some txt files in this section. (file1.txt, file2.txt, file3.txt, file4.txt. These files are located in biomedicine directory**. "NB. biomedicine directory is located in nimrworkshop**")
Then at the UNIX prompt, type, in our current working directory

```
$ cp ~/nimrworkshop/biomedicine/file1.txt .
```

Note: Don't forget the dot . at the end. Remember, in UNIX, the dot means the current directory.

The above command means copy the file **science.txt** to the current directory, keeping the name the same.

Exercise 2a

Create a backup of your **file1.txt** file by copying it to a file called **filexyz.txt**

2.2 Moving files

mv (move)

`mv` ***`file1 file2`*** moves (or renames) **file1** to **file2**

To move a file from one place to another, use the **mv** command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two.

It can also be used to rename a file, by moving the file to the same directory, but giving it a different name.

We are now going to move the file **filexyz.txt** to your **biomedicine** directory.

First, change directories to your **nimrworkshop** directory (can you remember how?). Then, inside the **nimrworkshop** directory, type

```
$ mv filexyz.txt biomedicine/.
```

Type ls and ls biomedicine to see if it has worked.

2.3 Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the `rm` command. As an example, we are going to create a copy of the **file1.txt** file then delete it.

Inside your **nimrworkshop** directory, type

```
$ cp file1.txt temporaryfile.txt
$ ls $ rm temporaryfile.txt
$ ls
```

You can use the `rmdir` command to remove a directory (make sure it is empty first). Try to remove the **biomedicine** directory. You will not be able to since UNIX will not let you remove a non-empty directory.

Exercise 2b

Create a directory called **temporarystuff** using `mkdir`, then remove it using the `rmdir` command.

2.4 Displaying the contents of a file on the screen

clear (clear screen)

Before you start the next section, you may like to clear the terminal window of the previous commands so the output of the following commands can be clearly understood.

At the prompt, type

```
$ clear
```

This will clear all text and leave you with the $ prompt at the top of the window.

cat (concatenate)

The command cat can be used to display the contents of a file on the screen. Type:

```
$ cat file1.txt
```

less

The command less writes the contents of a file onto the screen a page at a time. Type

```
$ less file1.txt
```

Press the [**space-bar**] if you want to see another page, and type [**q**] if you want to quit reading. As you can see, `less` is used in preference to `cat` for long files.

head

The **head** command writes the first ten lines of a file to the screen. First

clear the screen then type

```
$ head file1.txt
```

Then type

```
$ head -5 file1.txt
```

What difference did the -5 do to the head command?

tail

The **tail** command writes the last ten lines of a file to the screen. Clear

the screen and type

```
$ tail file1.txt
```

Q. How can you view the last 15 lines of the file?

2.5 Searching the contents of a file

Simple searching using less

Using **less**, you can search through a text file for a keyword (pattern). For example, to search through **science.txt** for the word **'science'**, type

```
$ less file1.txt
```

then, still in **less**, type a forward slash [**/**] followed by the word to search

```
/file1
```

As you can see, **less** finds and highlights the keyword. Type [**n**] to search for the next occurrence of the word.

grep (Global regular expression print)

grep is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

```
$ grep genomics file1.txt
```

As you can see, **`grep`** has printed out each line containing the word **science**.

Or has it ????

Try typing

```
$ grep Genomics file1.txt
```

The **`grep`** command is case sensitive; it distinguishes between Genomics and genomics.

To ignore upper/lower case distinctions, use the -i option, i.e. type

```
$ grep -i Genomics file1.txt
```

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

```
$ grep -i 'genome assembly' file1.txt
```

Some of the other options of grep are:

-v display those lines that do NOT match
-n precede each matching line with the line number **-c**
print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time. For example, the number of lines without the words science or Science is

```
$ grep -ivc genomics file1.txt
```

wc (word count)

A handy little utility is the **`wc`** command, short for word count. To do a word count on **science.txt**, type

```
$ wc -w file1.txt
```

To find out how many lines the file has, type

```
$ wc -l file1.txt
```

Summary

Command	Meaning
cp ***file1 file2***	copy file1 and call it file2
mv ***file1 file2***	move or rename file1 to file2
rm ***file***	remove a file
rmdir ***directory***	remove a directory
cat ***file***	display a file
less ***file***	display a file a page at a time
head ***file***	display the first few lines of a file
tail ***file***	display the last few lines of a file
grep ***'keyword' file***	search a file for keywords
wc ***file***	count number of lines/words/characters in file

UNIX Three

3.1 Redirection

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

We have already seen one use of the **`cat`** command to write the contents of a file to the screen. Now

type **`cat`** without specifying a file to read

```
$ cat
```

Then type a few words on the keyboard and press the [**Return**] key.

Finally hold the [**Ctrl**] key down and press [**d**] (written as **^D** for short) to end the input.

What has happened?

If you run the **`cat`** command without specifying a file to read, it reads the standard input (the keyboard), and on receiving the 'end of file' (**^D**), copies it to the standard output (the screen).

In UNIX, we can redirect both the input and the output of commands.

3.2 Redirecting the Output

We use the > symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of fruit, type

```
$ cat > list1
```

Then type in the names of some fruit. Press [**Return**] after each one.

```
pear banana
apple
^D {this means press [Ctrl] and [d] to stop}
```

What happens is the cat command reads the standard input (the keyboard) and the > redirects the output, which normally goes to the screen, into a file called **list1**

To read the contents of the file, type

```
$ cat list1
```

Exercise 3a

Using the above method, create another file called **list2** containing the following fruit: orange, plum, mango, grapefruit. Read the contents of **list2**

3.2.1 Appending to a file

The form >> appends standard output to a file. So to add more items to the file **list1**, type

```
$ cat >> list1
```

Then type in the names of more fruit

```
peach grape
orange
^D (Control D to stop)
```

To read the contents of the file, type

```
$ cat list1
```

You should now have two files. One contains six fruit, the other contains four fruit.

We will now use the cat command to join (concatenate) **list1** and **list2** into a new file called **biglist**. Type

```
$ cat list1 list2 > biglist
```

What this is doing is reading the contents of **list1** and **list2** in turn, then outputing the text to the file **biglist**

To read the contents of the new file, type

```
$ cat biglist
```

3.3 Redirecting the Input

We use the < symbol to redirect the input of a command.

The command sort alphabetically or numerically sorts a list. Type

```
$ sort
```

Then type in the names of some animals. Press [Return] after each one.

```
data
genome
gene
ape
file
out
^D (control d to stop)
```

The output will be

```
ape
data
file
gene
genome
out
```

Using < you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

```
$ sort < biglist
```

and the sorted list will be output to the screen. To

output the sorted list to a file, type,

```
$ sort < biglist > slist
```

Use cat to read the contents of the file **slist**

3.4 Pipes

To see who is on the system with you, type

```
$ who
```

One method to get a sorted list of names is to type,

```
$ who > names.txt $ sort < names.txt
```

This is a bit slow and you have to remember to remove the temporary file called names when you have finished. What you really want to do is connect the output of the who command directly to the input of the sort command. This is exactly what pipes do. The symbol for a pipe is the vertical bar |

For example, typing

```
$ who | sort
```

will give the same result as above, but quicker and cleaner. To

find out how many users are logged on, type

```
$ who | wc -l
```

Exercise 3b

Using pipes, display all lines of **list1** and **list2** containing the letter 'p', and sort the result.

Summary

Command	Meaning
`command > file`	redirect standard output to a file
`command >> file`	append standard output to a file
`command < file`	redirect standard input from a file
`command1 | command2`	pipe the output of command1 to the input of command2
`cat file1 file2 > file0`	concatenate file1 and file2 to file0
`sort`	sort data
`who`	list users currently logged in

Reference

1. http://easysimplelearning.com
2. http://www.ee.surrey.ac.uk/Teaching
3. H3ABioNet.org