Introduction to Unix

Tejal Joshi

Tutorial adopted from “Unix For Beginners”
by Michael Stonebank
http://www.ee.surrey.ac.uk/Teaching/Unix/
Introduction to the UNIX Operating System

- What is UNIX?
- Files and processes
- The Directory Structure
- Starting an UNIX terminal
- Working with UNIX commands
What is UNIX?

- An operating system developed in 1969 – began it’s life
- Under continuous developments since then
What is Unix

• Graphical user interface (GUI) is available, but knowledge of the UNIX user environment is required to carry out operations not covered by a GUI.
• secure shell (ssh) session.
Types of UNIX

- There are many different versions of UNIX, although they share basic similarities.
  - Sun Solaris,
  - GNU/Linux,
  - MacOS X (Darwin).
- We use GNU/Linux or IRIX® on our servers.
The UNIX operating system is made up of three parts; the kernel, the shell and the programs (applications).
The shell

• Interface between user and kernel.
• Example of shell’s work:

When you login, following happens:

1. check username and password
2. If successful, start shell (also a commandline interpreter)
3. For example, remove a file “deleteme”
   \textit{rm delete me}
4. Shell interprets and passes the command to the operating system
5. The file gets removed
Shell is customizable
Shell can complete your commands/file name when you press [TAB].
Command History

• The shell keeps a list of the commands you have typed in.
• If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.
Files and processes

Everything in UNIX is either a file or a process, even a computer display.

- A process is an executing program identified by a unique PID (process identifier).
- A file is a collection of data. They are created by users using text editors, running compilers etc.

Examples of files:
- a document (report, essay etc.)
- the text of a program written in some high-level programming language
- instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- a directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.
Enough of the theory, let’s get some hands on
UNIX directory structure

All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called root (written as a slash /)
Listing files and directories

Command `ls` lists the contents of your current working directory.

- `ls` ; shows a listing of files and directories
- `ls -a` ; also shows hidden files
- `ls -l` ; detailed listing of files and directories
- `ls -lt` ; time sorted listing
- `ls -lsh` ; detailed listing, with “human-readable” file sizes (GB, MB, KB, etc).
- `ls -l pathname` ; gives listing for a given directory
- `ls -l filename` : gives details on a specific file

More options available. Look for the help page.

> man ls
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 unixstuff in your current working directory type

```bash
> mkdir unixstuff
```

To see the directory you have just created, type

```bash
> ls
```
Changing Directories

**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

```plaintext
> cd unixstuff
```

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

**Exercise**

Make another directory inside the `unixstuff` directory called `backups`

Still in the `unixstuff` directory, type

```plaintext
> ls -a
```

As you can see, in the `unixstuff` directory (and in all other directories), there are two special directories called (`.`) and (`..`)
The current directory (.)

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

```bash
> cd .
```

NOTE: there is a space between cd and the dot

means stay where you are (the `unixstuff` 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

```bash
> cd ..
```

will take you one directory up the hierarchy (back to your home directory). Try it now.

Note: typing cd with no argument always returns you to your home directory. This is very useful if you are lost in the file system.
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/people/stud110`

which means that **stud110** (your home directory) is in the sub-directory **people** (the group directory), which in turn is in the **home** sub-directory, which is in the top-level root directory called " / ".
More about home directories and pathnames

Understanding pathnames

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

```
> ls unixstuff
```

to list the contents of your unixstuff directory.

Now type

```
> ls backups
```

You will get a message like this -

```
backups: No such file or directory
```

The reason is, **backups** 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 backups directory, you must type

```
> ls unixstuff/backups
```
~ (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

```bash
> ls ~/unixstuff
```

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

What do you think

```bash
> ls ~
```

would list?

What do you think

```bash
> ls ~/..
```

would list?
Summary

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
Copy files with the `cp` command:

```
cp (copy) file1 file2
```

First, `cd` to your `unixstuff` directory.

```
> cd ~/unixstuff
```

Then at the UNIX prompt, type,

```
> cp /home/people/tejal/public_html/courses/Random_text.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 `Random_text.txt` to the current directory, keeping the name the same.

**Exercise**

Create a backup of your `Random_text.txt` file by copying it to a file called `Random_text.bak`
**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 Random_text.bak to your backup directory.

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

> mv Random_text.bak backups

Type ls and ls backups to see if it has worked.
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 `Random_text.txt` file then delete it.

Inside your `unixstuff` directory, type

```
> cp Random_text.txt tempfile.txt
> ls
> rm tempfile.txt
> ls
```

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

**Exercise**

Create a directory called `tempstuff` using `mkdir`, then remove it using the `rmdir` command.
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 Random_text.txt
```

As you can see, the file is longer than than the size of the window, so it scrolls past making it unreadable.
less to view file contents

**less**

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

```bash
> less Random_text.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 Random_text.txt

Then type

> head –n 5 Random_text.txt

What difference did the –n 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 Random_text.txt

Q. How can you view the last 2 lines of the file?
Searching the contents of a file

**Simple searching using less**

Using less, you can search though a text file for a keyword (pattern). For example, to search through `Random_text.txt` for the word “HGNC”, type

```bash
> less Random_text.txt
```

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

```bash
/ENSG00000061273
```

As you can see, less finds and highlights the keyword. Type `[n]` to search for the next occurrence of the word.
Wildcard characters in file names

• What if you do not know the file name?
  • for example, find files with “txt” extension
    > ls *.txt
  • find files with 5 letter names starting with “gene” and extension .fa
    > ls gene?.fa

You can use wildcard characters with nearly all Unix commands, as long as they are not ambiguous or risky.
Summary on wildcards

* matches any number of characters in a filename, including none.

? matches any single character.

[ ] Enclose a set of characters, any one of which may match a single character at that position.

- Used within [ ] to denote a range of characters. E.g. [A-Z], [a-b], [1-6], etc.
Wildcard characters

- To copy all files from /home/people/tejal/public_html/courses to your home directory.
  
  \texttt{cp -R /home/people/tejal/public_html/courses/* .}

Exercise:

Find fasta files (.fa) starting with \textit{pou}, have at least 5 characters (including \textit{pou}).
Grep, pattern search tool

grep \textit{pattern} filename

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

```
> grep hgnc Random_text.txt
```

As you can see, \texttt{grep} has printed out each line containing the word \texttt{hgnc}.

Or has it ????

Try typing

```
> grep -i hgnc Random_text.txt
```

grep command is case sensitive; it distinguishes between hgnc and HGNC. Use \texttt{i} to ignore case.
Grep cont.

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 'hgnc' Random_text.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 hgnc or HGNC is

```
> grep -ivc hgnc Random_text.txt
```
Complex patterns can be shown as regular expressions

REGULAR EXPRESSIONS
Regular Expressions

• There are three operators used to build regular expressions. Let R and S be regular expressions and L(R) the set of strings that match R.

  • Union
    • R | S
      \[ L(R \mid S) = L(R) \cup L(S) \]
  • Concatenation
    • RS
      \[ L(RS) = \{ rs, r \in R \text{ and } s \in S \} \]
  • Closure
    • R*
      \[ L(R^*) = \{ \epsilon, R, RR, RRR, \ldots \} \]
Regular Expressions

• a | (ab)
• a
• a*b*
• (ab)*
• a | bc*d
• letter = a | b | c | ... | z | A | B | C | ... | Z | _
• digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
• letter(letter | digit)*
Regular Expressions

^ beginning of line
$ end of line
. any single character
[...] any one of the characters in ... ; E.g. [A-Z]
[^...] any single character not in ... ; ranges are legal.

\n what the nth character \((...)\) matched (grep only)
r* zero or more occurrences of regular expression r
r+ one or more occurrences of regular expression r
r1r2 regular expressions r1 followed by r2
^$ Empty lines

[[:space:]] Any space characters (white space, tab, etc.)
[[:alnum:]] Any alphanumeric (0-9 or A-Z)
A handy little utility is the `wc` command, short for word count. To do a word count on `Random_text.txt`, type

```
$ wc -w Random_text.txt
```

To find out how many lines the file has, type

```
$ wc -l Random_text.txt
```
Regular expression

• Use of regular expression is wide-spread.

• UNIX grep (grep –E), egrep and awk use regular expression.

• Perl and Python uses them too.
Practice more

- `grep -E` or `egrep` can be used with regular expressions.

- `grep -f patternfile file` will search list of patterns in the `patternfile` in `file`. 
Working with Files

• Redirection

• Redirecting the Output

• Redirecting the Input

• Pipes
Redirection

• UNIX commands mostly write output to standard out (screen).
• Most commands take input from standard input (keyboard)
• Cat command without filename:

> 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?

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

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

We use the < symbol to redirect the input of a command. Using < you can redirect the input from a and 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
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.

Can this be done without creating a temporary file?

Yes, use pipe ( symbol “|” ) [Vertical bar]

> 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

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

Answer

> cat list1 list2 | grep p | sort
## Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>command &gt; file</code></td>
<td>redirect standard output to a file</td>
</tr>
<tr>
<td><code>command &gt;&gt; file</code></td>
<td>append standard output to a file</td>
</tr>
<tr>
<td><code>command &lt; file</code></td>
<td>redirect standard input from a file</td>
</tr>
<tr>
<td>`command1</td>
<td>command2`</td>
</tr>
<tr>
<td><code>cat file1 file2 &gt; file0</code></td>
<td>concatenate file1 and file2 to file0</td>
</tr>
<tr>
<td><code>sort</code></td>
<td>sort data</td>
</tr>
<tr>
<td><code>who</code></td>
<td>list users currently logged in</td>
</tr>
</tbody>
</table>
Data manipulation

• Replace a character to another character

• for example,

> cat Random_text.txt | tr ‘c’ ‘d’ ; will translate “c” into “d”

tr takes input from the standard out only.
Data manipulation

• suppress multiple occurrences of a character, for example a white space

> cat Random_text.txt | tr -s " "
Finding a specific column of a file

- `cut –f fieldnumbers –d delimiter filename`

- `cut –f 1 –d “;” evalSet`
  
  will print first column of evalSet file, columns are delimited by “;”.

- `cut –f 1-2 –d “;” evalSet`
  
  will print find a range of columns separated by “;”