Student and staff IT introduction

A very brief introduction to UNIX

Pierre-Yves Dupont - p.y.dupont@massey.ac.nz - ssiti2013@gmail.com

1 Introduction

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X. There are several distributions of Linux: Ubuntu, Debian, Fedora,... All UNIX commands should have the same behavior on all UNIX machines.

2 The Shell

The shell is a command line interpreter (CLI). It interprets the commands the types and displays the results. The commands are themselves programs. The shell allows filename and command name completion using TAB key. The shell keeps a list of the commands you have typed in. This list is given by the command *history*.

3 Simple commands

3.1 Directory Structure

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

3.1.1 ls (list)

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, and it is where your personal files and subdirectories are saved. To find out what is in your home directory, type

```
ls
```

The *ls* command (lowercase L and lowercase S) lists the contents of your current working directory. By default, this command doesn't display hidden files and directories. Files and directories having a name starting by a dot (.) are considered as hidden files on UNIX systems. 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
```

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. To access to the documentation of most of UNIX commands type

```
man ls
```

This command displays the manual of *ls*.

Remark: The current directory is represented by one dot (.), the parent directory by two dots (..) and your home directory by a tilde ( ).

3.1.2 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 *ssiti* in your current working directory type

```
mkdir ssiti
```

3.1.3 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 ssiti
```

To go in the parent directory type

```
cd ..
```

To go to the root of the system, type

```
cd /
```

To go back to your home directory, type

```
cd
```

The command *cd* without any argument returns you to your home directory.
To know where you are in the directories structure, type

```
pwd
```

The command *pwd* displays the *p*ath to the *w*orking *d*irectory.

3.2 Play with files and directories

3.2.1 Create, copy, rename and move

To create and empty file, type

```
touch my_new_file
```

Remark: it's better to avoid to use spaces in file or directory names. You can replace them by underscores (_)

To copy a file, type

```
cp file1 file2
```

This command makes a copy of file1 in the current working directory and calls it file2.
To copy a directory, type

```
cp -r dir1 dir2
```

To move a file, type

```
mv file1 file2
```

The syntax is really close to *cp* but 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.

To remove a file, type

```
rm file1
```

To remove a directory, you can use

```
rm -r directory
```

With the *-r* option *rm* will remove all files in the directory then the directory. Be careful!!

3.2.2 Display text files

To look at the contents of a file, you can use

```
cat file1.txt
```

This command shows the entire file. That's fine for an 8 KB file, but you wouldn't want to use this command on a 5 GB file!

To look at the file in pieces, use

```
less file1.txt
```

In the *less* program, you can move using the arrows. Use 'q' to quit. Some useful options for *less* command:

- *-S*: Causes lines longer than the screen width to be chopped rather than folded
- *-N*: Causes a line number to be displayed at the beginning of each line in the display
- hit *slash* in less environment to search a word (n and N keys to find next and previous occurrence)
- enter a line number in less environment to go to the given line

If the file is really big, it may be better to use the *head* and *tail* commands.

```
head -n 8 file1.txt
tail -n 8 file1.txt
```

head -n 8 displays the 8 first lines of the file. *tail -n 8* displays the 8 last lines.

It's possible to combine the two commands

```
head -n 50 test_fasta_file.fna | tail -n 10
```

Using this command, you will display the lines 40 to 50 of your file. The pipe (|) character is used to redirect the output of the first command (*head*) to the second (*tail*).

3.2.3 More useful commands on files

Counting the number of lines in a file:

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

wc command can be also used to count the number of characters in a file:

```
wc -m file1.txt
```

and also to count the number words:

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

To find the lines containing a pattern:

```
grep '>' file1.txt
```

This command will return all the lines of the file containing the pattern '>'. By default, the search is case sensitive, to ignore the case, use *-i* option.
To count the number of line containing a pattern:

```
grep -c '>' file1.txt
```

Here, the command returns the number of line containing the character '>'. To count the number of '>' in the file, you can use:

```
grep -o '>' file1.txt | wc -l
```

If you are interested in having only the number of lines starting by '>', use the following command:

```
grep -P -c  '^>' file1.txt
```

The sign '^' means 'beginning of the line'. If you want to find the number of lines ending by a dot (.) use the following command:

```
grep -P -c '\.$' file1.txt
```

The sign '$' means 'end of the line'. The dot is a special character for grep meaning 'any alphanumeric character', so if you are looking for a real dot, you have to *escape* it using the backslash (\) sign.

3.2.4 Output Redirection

On UNIX it's possible to redirect the output of a command to a file. To do that we use > sign.

```
ls > myfile.txt
```

This command writes the results of the *ls* command in the *myfile.txt*. If the file doesn't exists, it will be created, if it exists it will be erased and recreated. It's also possible to append a the end of a file:

```
ls >> myfile.txt
```

3.2.5 CSV and Tab delimited files

UNIX is really efficient to work on files containing columns of data delimited by a separator (tabulation, comma...). For example, it's possible to sort a large file given a column in some seconds.

```
1 sort -k 2 -n file1.txt
2 sort -k 2 -n -r file1.txt
```

The first line correspond to a sort of a space delimited file (default format for *sort* command) on the second column using a numeric sort. The line two correspond to a reverse sort of the file. To use *sort* command on CSV files (delimited by a comma), you must specify a delimiter:

```
sort -k 2 -n -t$',' file2.csv
```

It is also possible to sort on more than one column:

```
sort -k1,1n -k2,2 -t$',' file2.csv
```

Here *file2.csv* will be sorted first on the first column (*-k1,1n*) using a numeric sort (*n*) then on the second column (*-k2,2*).
Sometime it can be useful to make a sort on a file ignoring the first line (header line). To do that you must combine three commands:

```
(head -1 file.csv ; tail -n +2 file.csv | sort -t$',' -k1,1 -k2,2n) > file2.csv
```

tail -n +2 file.csv displays all the lines of the file ignoring the first line. The result of that command is used in the sort (*sort -t$',' -k1,1 -k2,2n*). To print the header line in the result file (*file2.csv*) you have to execute "*head -1 file.csv*" command which displays only the first row of the file.

It is also easy to extract columns from a file:

```
cut -f2-4,6 -d$'\t' file1.txt
```

This command extracts the columns 2 to 4 (2-4) and 6 (,6) in a tab delimited file.
It is also possible to display huge delimited files (impossible to display in MS-Excel or equivalent software) using the *column* command.

```
column -s $'\t' -t -n < file.txt
```

In this command, the option *-s* allows to set a delimiter character, the option *-t* ask to the command to determine automatically how to display the result and *-n* prevent the command to merge the consecutive delimiters.
It is also possible to find unique sequential entries in a file using;

```
uniq file.txt
```

You can use the option *-c* to only count the number of unique sequential lines in a file.
If you want to change the delimiter of a file you can entrer:

```
sed -i 's/;/\t/g' file.txt
```

This command will substitute (*s*) all (*g* global) ';' by a tabulation '\t' in the given file. It replace the file, so be sure to make a copy first. It is also possible to write:

```
tr ';' '\t' < file.txt
```

It is also possible to join two files. First "horizontally":

```
paste -d$'\t' test_file1.txt test_file2.txt
```

This command will produce a file containing the columns from the first file then the columns from the second. Take care of the order first.

```
join -j1 -t$'\t' test_file1.txt test_file2.txt
```

This command will join the two files using the first column as reference (*-j1*). If two values are identical in the reference column, *join* will concatenate the values from the two files. The rows from one file which are not in the other will be ignored.
Finally it's also possible to join two files 'vertically' using the command cat:

```
cat test_file1.txt test_file2.txt
```

4 Text file edition

4.1 Microsoft Word

Microsoft Word (and its free versions like OpenOffice and LibreOffice) are not considered as text editors but as word processors. A word processor is a computer application used for the production (including composition, editing, formatting and possibly printing) of any sort of printable material. These software append hidden characters to the text. These characters may be copied when you copy and paste the text!

The size of pure text file containing only the sentence "Hello World" is 12bytes. The size of a *docx* document containing exactly the same sentence is 3.4KB (283 times more than the text file!).

If you want to work on pure text files, you must use simple text editors. Text editors can be Smultron or Text Wrangler on MacOS; Wordpad or Notepad++ on MS Windows and Vi, emacs or nano on UNIX.

Remark. Emacs, Vi and nano are available also on MS Windows and MacOS.

4.2 Nano

The easiest text editor on UNIX is nano. This editor is inside the shell, there is no graphical interface, so it is possible to use even if you don't have access to a graphical session (for example on all computation servers). To open a file using nano, type:

```
nano file.txt
```

All available commands are displayed at the bottom of the screen. '$\hat{}G$' means *Control+g*. This command display the help of nano. To save, use '$\hat{}O$', to exit use '$\hat{}X$'.

Remark The Undo/Redo function are, most of the time, not available. On some UNIX and Linux distributions, it is possible to give a *-u* option allowing to have an access to theses functions using M-U (understand Meta-U aka ESC-U) to undo and M-E to redo.

5 Linux Cheat Sheet

For a good UNIX cheat sheet, you can go to `http://www.cheatography.com/davechild/cheat-sheets/linux-command-line/` For a very complete list of commands, you can have a look to the UNIX Toolbox reference website: `http://cb.vu/unixtoolbox.pdf`

A simple cheat sheet on next page

Unix/Linux Command Reference

FOSSwire.com

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command* | grep *pattern – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

Installation

Install from source:
./configure
make
make install
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

* use with extreme caution.