Introduction to Unix

We will be using the Linux operating system throughout this module. Linux is a freely available operating system which is comprised of open-source software. There are very many different variants of Linux, called distributions or distros for short. Each distribution packages a different subset of applications and/or user-interfaces, but all of them share the same kernel. The distribution we will be using in the lab sessions is called CentOS.
However, if you want to run Linux at home or on your laptop, the software we will be using in the course will run on most other distributions, including the popular Ubuntu or KUbuntu distributions. It is also possible to use MySQL from Windows. However, some of the commands for processing files are only available in Linux.

1 Introduction to Linux

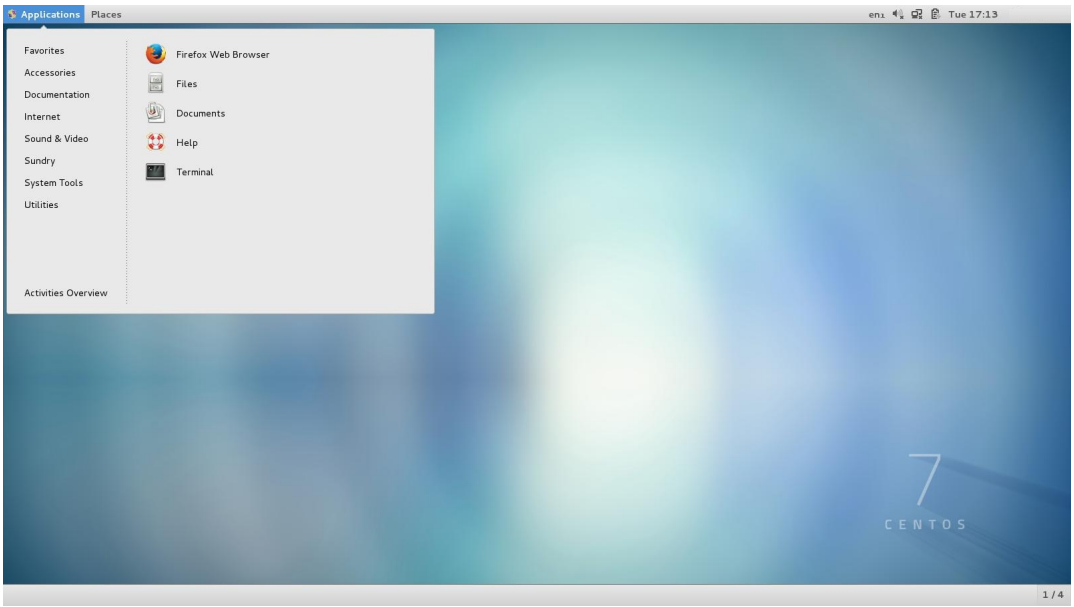
1. *Boot into the Linux Operating System*

   The machines in the lab are set up to run either Windows or Linux. If your workstation is not currently running Linux, then you will need to restart it in order to load the Linux operating system; on the first screen that appears when the machine is rebooting, choose CentOS Linux from the menu.

   Once Linux has booted, you should see a welcome screen with a login prompt.

2. *Log in*

   Now you can log in using your informatics id and password. Your informatics id should be the same as your k number. Once you have logged in you should see your the CentOS desktop:



   If you have used Windows before, it should look very familiar. It has a menu button in the top-left hand corner of the screen which plays a similar role to the Windows "start" menu; from here you can launch applications. On the bottom of the desktop is the task bar, which shows the active applications.

2 Using the Command-line

One of the powerful aspects of Linux operating system is that everything that can be done with the graphical user-interface can also be accomplished by typing commands. This has the advantage that any task can easily be scripted and automated.
When we type commands we are interacting with an application called the Unix shell. The shell is a simple interpreter which processes commands and manages their output. We typically interact with the shell using a console, which is a simply a windowed application which takes commands from the keyboard and shows output inside the window.

1. Launch the application called console from the Applications menu ($Applications \rightarrow Utilities \rightarrow Terminal$).

3 Listing files and directories (`ls`)

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, k123456, 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 lists the contents of your current working directory. There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.
To get a more detailed listing use the option `-l`:

```
ls -l
```

`ls` is an example of a command which can take options: `-l` 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. Typically, you can also find out what the particular options are for a command by using the special option `--help`

4 Making Directories (`mkdir`)

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

```
mkdir unixstuff
```

To see the directory you have just created, type:

```
ls
```

4.1 Changing to a different directory (`cd`)

The command `cd` (change directory) 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 unixstuff
```

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

Exercise 1. *Make another directory inside the* ***unixstuff*** *directory called* ***backups***

4.2 The directories . and ..

Still in the `unixstuff` directory, type:

```
ls -a
```

Option `-a` lists files that are normally hidden. As you can see, in the `unixstuff` directory (and in all other directories), there are two special directories called (`.`) and (`..`)

4.2.1 The current directory (.)

In Linux, (`.`) means the current directory, so typing:

```
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.

4.2.2 The parent directory (..)

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

```
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.

5 Pathnames (pwd)

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:

```
/mnt/st/k12345
```

which means that `k12345` (your home directory) is in the sub-directory `st` (the group directory), which is in the `mnt` sub-directory, which is in the top-level root directory called `/` .

Exercise 2. *Use the commands* ***cd****,* ***ls*** *and* ***pwd*** *to explore the file system.*

(Remember, if you get lost, type `cd` by itself to return to your home-directory)

5.1 More about home directories and pathnames

5.1.1 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
```

5.1.2 ~ (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 ~/unixstuff
```

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

6 Tab completion & Command Line History

6.1 Tab completion

Tab completion is an extremely helpful feature in nearly any command-line environment. Just hit Tab while typing a command, option, or file name and the shell environment will automatically complete what youre typing or suggest options to you.
For example, let's say you want to list the contents of your `unixstuff` directory. You can just type:

```
ls uni
```

and press Tab — if your system doesn't have any other files or directories that begin with those letters, Bash will automatically fill in `unixstuff` and you can press Enter to run the command.
Tab completion is especially useful when typing file names, directories, and paths. Rather than trying to type a long file name, you can just start typing the beginning of the name and press Tab.

6.2 Command Line History

Another useful utility when typing commands is the **Command Line History**. You can use the up and down key's to scroll through previously typed commands. Press Enter to execute them or use the left and right arrow keys to edit the command first.

Exercise 3. *Right, now let's put this stuff into practice. Have a go at the following:*

- *Let's start by getting familiar with moving around. Use the commands cd and ls to explore what directories are on your system and what's in them. Make sure you use a variety of relative and absolute paths. Some interesting places to look at are:*
  - */etc - Stores config files for the system.*
  - */var/log - Stores log files for various system programs. (You may not have permission to look at everything in this directory. Don't let that stop you exploring though. A few error messages never hurt anyone.)*
  - */bin - The location of several commonly used programs (some of which we will learn about in the rest of this tutorial.*
  - */usr/bin - Another location for programs on the system.*
- *Now go to your home directory using 3 different methods.*
- *Make sure you are using Tab Completion when typing out your paths too. Why do anything you can get the computer to do for you?*

7 Copying Files (cp)

`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 create anew file, and use the cp command to copy it to your `unixstuff` directory.
First, cd to your home directory:

```
cd
```

Create a new file named `science.txt` as follows:

```
base64 /dev/urandom | head -c 10000 > science.txt
```

Then cd to `unixstuff` directory:

```
cd ~/unixstuff
```

Then at the UNIX prompt, type:

```
cp ~/science.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 4. *Create a backup of your* `science.txt` *file by copying it to a file called science.bak*

8 Moving files (mv)

`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 `science.bak` to your backup directory.
First, change directories to your `unixstuff` directory (can you remember how?). Then, inside the `unixstuff` directory, type:

```
mv science.bak backups/.
```

Type `ls` and `ls backups` to see if it has worked.

9 Removing files and directories

To delete (remove) a file, use the `rm` command. As an example, we are going to create a copy of the `science.txt` file then delete it.
Inside your `unixstuff` directory, type:

```
cp science.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. When `rm` is run with the `-r` option it allows us to remove directories and all files and directories contained within.

Exercise 5. *We now have at our disposal, various commands to actually interact with the system. Let's put them into practice. Have a go at the following:*

- *Start by creating a directory in your home directory in which to experiment.*
- *In that directory, create a series of files and directories (and files and directories in those directories).*
- *Now rename a few of those files and directories.*
- *Delete one of the directories that has other files and directories in them.*
- *Move back to your home directory and from there copy a file from one of your subdirectories into the initial directory you created.*
- *Now move that file back into another directory.*
- *Rename a few files*
- *Next, move a file and rename it in the process.*

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

```
cat science.txt
```

As you can see, the file is longer than than the size of the window, so it scrolls past making it unreadable.
The command `less` writes the contents of a file onto the screen a page at a time. Type:

```
less science.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.
The `head` command writes the first ten lines of a file to the screen.
First clear the screen then type:

```
head science.txt
```

Then type:

```
head -5 science.txt
```

The `tail` command writes the last ten lines of a file to the screen.
Clear the screen and type:

```
tail science.txt
```

11 Searching the contents of a file

Using less, you can search though a text file for a keyword (pattern). Create a new file as follows:

```
ls -la > file.txt
```

For example, to search through `file.txt` for the word '.txt', type:

```
less file.txt
```

then, still in less, type a forward slash `/` followed by the word to search '.txt', as follows:

```
/.txt
```

As you can see, less finds and highlights the keyword. Type `n` to search for the next occurrence of the word.
`grep` is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type:

```
grep .txt file.txt
```

As you can see, `grep` has printed out each line containing '.txt'. The `grep` command is case sensitive. To ignore upper/lower case distinctions, use the `-i` option. 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

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

```
wc -w file.txt
```

To find out how many lines the file has, type:

```
wc -l file.txt
```

12 SSH

Secure Shell (SSH), sometimes known as Secure Socket Shell, is a UNIX-based command interface and protocol for securely getting access to a remote computer. It is widely used by network administrators to control servers remotely. This is protocol is also used for accessing KCL High Performance Cluster or to access the NMS's database server remotely. The syntax of the ssh login command is as follows:

```
ssh username@server
```

Once you have connected to the server, you will probably be asked to verify your identity by providing a password.
The Informatics department has a server named "calcium" that can be used to access other server that cannot be accessed from outside KCL network (such as the NMS MySQL server). To log on and connect to calcium you can use the following command (where k123456 is your username):

```
ssh k123456@calcium.inf.kcl.ac.uk
```

It is possible that your have to confirm that you want to connect to the host by typing 'yes':

```
Are you sure you want to continue connecting (yes/no)? yes
```

To exit back into your local session, simply type:

```
exit
```

13 Personal Study

If you finish these activities early then you can start this personal study in the lab, otherwise you should be working on these tasks in your own time.

1. There are many other useful commands. Take a look on some of the existing Unix tutorials:

   http://www.tutorialspoint.com/unix/