Linux Introduction

A large amount of computational programs are available on Linux operating system (OS), and it is the system of choice for our computational department. This booklet should act as an introduction to Linux, covering the very basics on how to start using it effectively. It discusses the very basics, and includes questions for you to answer.

This is NOT part of an assessment - it is purely for your benefit to make good use of the rest of the dry labs and computational projects (if you have one). Make notes as you go along if you wish - these booklets will be for you to keep.

Please hand in at the end of the session to get some feedback on the questions.

Outline:

- Introduction
- Exercise 1 - getting familiar with the system
- Exercise 2 - getting around
- Exercise 3 - creating and removing directories
- Exercise 4 - dealing with files
- Exercise 5 - text editors
- Exercise 6 - execution of the programs from the terminal
- Exercise 7 - permissions
- Exercise 8 - unzipping and untarring
- True or False

Introduction

The session is divided into several exercises, each of which concentrates on a particular aspect of Linux. Within each exercise you'll find a short introduction/explanation, followed by some questions regarding the topic. As you work through the content you will use concepts learned in the previous exercises. Each section will be explained and plenty of help will be available, however it's up to you to work through the content.

At the end of this dry lab you should:

- know how to use the terminal
- be familiar with common shortcuts
- be able to find your way around a linux system
- know how to create and remove files and folders in the terminal
- know how to use some text editors

Exercise 1 - getting familiar with the system:

Linux can come in different flavours - known as distributions (or distros) such as CentOS, openSUSE, Ubuntu, Mint, Arch etc. Regardless of the distro, the system consists of a shell (basic user interface) as well as the graphical user interface (gui). Within the gui you can access the shell via the terminal (aka command line). The terminal is extremely important to efficiently use the system - the terminal and the gui are complimentary to each other.

At the very top we have the root directory (dir) which encompasses the entire system. The space is divided into directories the system uses - one of them is the user space (home) where different users and their files are stored.

Questions:

1. Open 'Details'. What version of Linux are you using?
2. Where is your user space located?
3. What is at the very top of the structure?

Exercise 2 - getting around:

Open the terminal. Now that you have the terminal open we can start getting around the system. There are a few commands and flags you will have to get familiar with to be able to do that:

Commands:
`pwd` - print working directory
`ls` - list folder content e.g. `ls [options] [location]`
`cd` - change directory

Flags (sometimes long versions of flags are used with - -):
`-l` - long
`-a` - all
`-v` - verbose

`-r` - recursive (or reverse)
`-i` - interactive

If you need help with the accepted flags in a program, you can always use `--help`:
`vi --help`

Paths:
The terminal works on absolute and relative paths: Absolute path is in relation to the root dir, and will start with /
Relative path will in relation to where you are now for example:
absolute : `ls /home/user/Documents`
relative : `ls Documents`

When using paths there are some commonly used shortcuts for different places:

`/` - root
`~` - home
`.` - current dir
`..` - parent dir

`cd` command is used to move around the system. You would use the absolute or relative path along with cd to change the folder
e.g. `cd /home/user/Documents`

To make life easier pressing 'tab' will auto-complete the path you tried to enter.

Try to go around some folders to get a feel for it and then answer the questions below.

Questions:

1. What is short for 'home'?
2. What is short for parent dir?
3. Move to the Documents folder using `cd`. Check where you are using `pwd` and write it below:
4. What are 2 different ways of checking the contents of the `exercise1` directory within the folder `drylabs_linux` (using `ls`) without leaving your current location? (think about paths and shortcuts)
5. What happens if you run `cd` without any flags?

6. Run `cd` with no flags and then with the 'all' flag - what difference do you notice?

7. What are the 4 different ways you can go to your home dir? List them below:

Exercise 3 - creating and removing directories:

Now that you know how to go around the folders and check their contents you might want to start dealing with creation and deletion of the folders. Here are some commands necessary for this:

`mkdir` - create a folder
`rmdir` - remove a folder

NOTE `rmdir` is final - no undoing. no retrieving. Also to delete the folder, it must be empty.

Questions:

1. Move to folder `exercise3`. Create a new directory `my_folder` and move into it. Run `pwd` and write the path below:

2. Move back to your parent folder. Remove the directory `my_folder`. Type `ls` - what does it show?

Exercise 4 - dealing with files:

Now in addition to creating and deleting folders, we can do the same with files (and folders are a type of a file). Unlike on Windows, Linux doesn't need an extension to be able to deal with the files. One other thing to remember is that Linux is case sensitive meaning that e.g. `file.txt` is different to `File.txt`.

Spaces tend to be the enemies in file names - a space is a divider between items, meaning that a folder called `Dry Labs` would be seen as two different items - `Dry` and `Labs`.

If you want to view the file you created you can use the command `more` and scroll through it using the spacebar. Specifically you can also look at the beginning and the end of a file using `head` and `tail` respectively. With these two commands you have to specify how many lines you'd like to view:
e.g. `head -3 file1` will display first 3 lines of `file1`.

If you would like to delete many files at the same time which have something in common in the name you can use a wildcard. `*` is a wildcard representing all characters.

Summary of some commands:
`more` - view the file
`head` - view start of file
`tail` - view end of file
`touch` - create a file
`rm` - remove file

Questions:

Move to folder `exercise4`:

1. Create a blank file called `file_one` using `touch`. Run `ls` - what does it say?

2. How do you remove a file? Write the FULL command below:

3. Try to remove the folder `to_delete` using `rm`. What happens?

4. How would you delete a folder using `rm`? Write the command below. (hint: look at exercise 2 flags)

5. Delete file `delete_me` using the flag `-i`. What happens? How do you delete an item using this flag?

6. What are some common file extensions you are familiar with? (e.g. `.txt`) (list 5):

7. What do the extensions you listed represent? (e.g. `.txt` is a text file):

8. What happens when you try to move into the folder `Dry Labs`?

9. How would you avoid spaces in file names? List some ideas (aim for 2-3):

10. How would you open a file which contains spaces? List two different methods:
    1.

    2.

11. How are hidden files shown as? (hint: remember to use flags)

12. Using `more`, what is the content of `example_file`? Write it below:

13. Look at `1zni.pdb` file.
    (a) what do the first 4 lines show?

    (b) what do the last 2 lines say?

14. Go into `logs_folder`. What is the content of that folder?

15. Using `ls a*` - what do you see?

16. How would you remove all files starting with `#` all at the same time? - Write the command below:

17. How would you remove all files with `.txt` extension? Write the command below below:

Exercise 5 - text editors:

To edit any type of a text file you use a text editor. Some of the popular ones are vi, nano, gedit and emacs - once you get familiar with them you can pick whichever one you feel most comfortable with. To open a file using a text editor you type the name of the editor followed by the filename e.g.:

```
vi file1
```

If the file you're trying to open doesn't exist, the editor will create (temporarily) a file with the name you specified. However the file will need to be saved to be actually created.

When you open vi you're in view mode.

To start using vi:

`i` - insert mode
`Esc` - exit the mode you're in, back into view mode
`:w` - save
`:wq` - save and exit
`:q!` - exit, no save

Using nano is similar to vi but the keyboard commands are different. When nano opens the file is ready to edit. Note all the different commands at the bottom of the terminal. To exit click `ctrl-x` - it will prompt for saving.

Exercise:

1. using vi (and the example above) create/open file `file_one`.
2. Press `i` to start editing. Write "Created using vi".
3. Save and exit.
4. create `file_two` using nano.
5. Write "This was created using nano"
6. Save and exit.

Once you finish this, ask for a demonstrator to have a look.

Exercise 6 - execution of the programs from the terminal:

You can execute (call) any program you like (Chimera, Gromacs, web browser, etc) from the terminal. Some programs can be launched from either a terminal or from an icon (Windows style) – such as Chimera of Firefox. For others – like Gromacs or Xmgrace – you need to use the terminal.

Exercise:

1. Launch UCSF Chimera from the terminal by typing:
   `chimera &`

   The symbol enables you to still use the terminal while the program is running. If you type:
   `chimera`

   you will still launch the program, but the terminal will be blocked – until you close the program.

   To save yourself some time, you can type...
   `chim`

   ... and hit tab

2. Launch the Gromacs menu:

   In the terminal window, type ...
   `gmx help commands`

   ...and you should see a list of Gromacs tools

   To get more information about a tool you can use the `-h` flag (help), or the 'help' command:
   ```
   gmx <tool> -h
   gmx help <tool>
   ```

   For example:
   `gmx pdb2gmx -h`

3. Launch Xmgrace and plot your data:

   Go to `exercise 6` folder:
   Launch...

   `xmgrace RMSF_ahr_res.xvg`

Exercise 7 - permissions:

Each file and folder will have a set of permissions associated with it, these include:

`r` - read
`w` - write
`x` - execute

The permissions are assigned separately for the owner, group and others.

**Note that `d` signifies a directory.

The permissions can be also edited and managed, however this will not be covered here as it is not needed. As you will not have the sudo password (admin password), if you need anything installed you can ask one of the PhDs/staff.

Questions:

Go to `exercise7` folder:

1. Use `ls -l`. What permissions are associated with `user_folder`? Write all the permissions below:

2. What permissions does the group have?

Exercise 8 - creating an archive and unzipping/untarring

In cases where you would like to pass files and folders along (distribute them), you would compress the files and create a single archive file out of them. In Linux an archive is generally a tarball (`.tar` file) whereas in Windows it's a `.zip` file.

The generally used flags in creating and unpacking an archive are:

`v` - verbose (show what the program is doing)
`z` - used for gzip files
`j` - used for bzip2 files
`f` - signifies a filename
`c` - create an archive
`x` - asks `tar` to extract the files

In order to create a tarball, you would use the `tar` command along with appropriate flags e.g.:

```
tar cvzf my_tarball.tar file1 folder1
```

In this case we are creating tarball called `my_tarball` containing `file1` and `folder1`

When you have an archive file, the files you'd like to use are compressed and you need to unpack them beforehand. To untar you use the `tar` command along with appropriate flags e.g.:

```
tar xvzf file.tar.gz
```

Exercise:

Go to `exercise8` folder:

1. Create a folder called `example_work`. Inside that folder create a new text file.
2. Create a gzip type archive of the folder you just created AND the file `add_me.log`
3. Which flags have you used to create this archive? Write them below:
4. Locate `tarball.tar.gz` (look through the folders)
5. Untar this archive using the appropriate flags.
6. Write below the flags you have used to decompress the archive:

Circle true or false:

1. Linux is extensionless **True** **False**
2. `my_protein.txt` is the same as `my_Protein.txt` **True** **False**
3. `-a` is the same as `--all` **True** **False**
4. os is installed in root **True** **False**
5. Actions in terminal (such as deleting) are reversible **True** **False**

Extra questions:

1. Why can't you see the password show up in terminal?
2. How do you browse terminal history?
3. How do you quit a program in terminal

Extra things

- Middle mouse button can be used for copying and inserting a highlighted text
- `ctrl-c` is used for quitting a process/program.

There are many more things associated with efficient use of Linux e.g bash scripting. If you're interested in more in depth Linux sessions let us know and we can organise something.