PHYSICS 2T C Programming Under Linux

Linux Lab 1 An introduction to the command line

INTRODUCTION

This lab will serve as a basic introduction to the command line using BASH – the Bourne Again SHell.

Once you have finished this lab you should be relatively comfortable using the command line for basic file management.

HOW TO ANSWER THE QUESTIONS

At the end of each section there are a set of questions, some of which will ask you about the tasks you have already completed in the lab and others will require you to do other research. Do not be put off if the answer is not obvious, read the Getting Help section if you are stuck.

Questions for this lab should be answered in `answers.txt` which can be found in the `Lab01` folder.

GETTING STARTED

Start by opening the Terminal application. If you are using Gnome this can be found under Applications → Accessories → Terminal. This will gave you access to a BASH shell which will allow you to enter commands.

Verify that you are in your home directory, you can do this by typing **`pwd`** (Print Working Directory) into the terminal. It should be similar to `/home/0800890`.

Next, enter `ls`. You will be present with a list of the visible files and folders in the current directory.

GETTING HELP

You can find more information about the **`pwd`** and `ls` commands using their manual pages. All of the labs can be completed using the information provided in the manual pages. As such, it is essential that you get used to using them.

Manual pages can be accessed via `man command`, where `command` is replaced by the name of command you wish to know more about. Take a quick look at the manual pages for BASH (`man bash`) and ls (`man ls`). You can exit the manual pages by pressing `q`.

It is possible to search for man pages using the `man -k` and `apropos` commands. `man -k delete` and `apropos delete` will both search for manual pages which have a description containing the word “delete”.

Manual pages exist for more than just user commands, `man intro`, which serves as a basic introduction to the command line and bash, is one example.

As more commands are introduced, you should read their particular man pages and get an idea of what the command can do. Reading the man pages and using the commands is by the far the most effective way of learning bash.

In addition, this lab comes with `Commands.pdf` which is a list of the most common commands and their most useful options. Try some of them out before you continue.

If you are stuck please remember to ask the demonstrators who will be available during labs.

You are, of course, allowed to use lecture notes, books and the internet while completing the labs.

FILES AND DIRECTORIES

You are going to create a basic directory structure which could be used for storying data and documents for an experiment. Some large experiments result in hundreds or even thousands of files being created – sorting and storing them in an organised manner is essential.

1. Assure that you are in your home directory using the `pwd` command.
   If you find that you are elsewhere then the command `cd` without any arguments will take you to your home directory.
2. Create a directory called `experiment`. If you are unsure how to do this then consult `Commands.pdf` or `man mkdir`. The new directory should be listed when you run `ls`.
3. Use the `cd` command to change your working directory to the `experiment` directory.
4. Create 3 new directories named `docs`, `data`, and `results` in the experiments directory. You should finish with the following directory structure which can be verified using `ls -R`.
   ```
   ~/experiment/
   ~/experiment/data
   ~/experiment/docs
   ~/experiment/results
   ```
5. Use the `cd` command to navigate back to your home folder then the `Lab01` directory.
6. You are going to run a BASH script which will generate some sample data files. Use `ls` to check for a file called `generate.sh` and run the script using `./generate.sh`. It may take a few seconds to finish running, when it does you can use the `less` and `more` commands to take a look at the source of the script.
7. Without changing directory, check the contents of the `experiment` directory using `ls ~/experiment`. The "~" will be automatically substituted by the absolute path of your home directory (You can see this in action using `echo ~`). The `experiment` directory should now contain 50 `.data` files.
8. Navigate back to the `experiment` directory. You need to move the data files into the `data` directory. Start by typing `mv 2` **but do not run the command yet.** Press the tab key twice. This will give you a list of all the files in the directly begging with 2. Press 4 and then tab again. This will complete the rest of the file name. Do the same again to add the name of the destination directory so that the entire command is `mv 24.data data/`.
   Check that the file has been moved using `ls data/`.
9. Moving each file individually would be slow. There is a much quicker way of selecting all `.data` files using the * character. Run the command `mv *.data data/`. The * will match any number of any characters (including nothing or null) so that any file which ends in `.data`

will be selected.
Change into the `data` directory and use `ls` to verify all the files have been moves.

10. You want to filter some specific data files to look at later. Create a new directory called `5s` and copy, using the `cp` command, any file which has a name containing 5 into it. *Remember you can use the * character to move multiple files at once.*
11. The plan is to perform several experiments over the new few weeks. As such, you want to store them all in an organised manner. Create a new directory called `experiments` in your home directory which will store all of the experiments and their data.
12. Copy the `experiment` directory into the `experiments` directory, renaming it to `exp1` (You will need to use the `-R` argument to copy a directory). You should only need a single command to do this.
13. Try to remove the old `experiment` directory using the `rmdir` command.
14. Use the `rm` command to remove the directory. Check its man page or `Commands.pdf` for how to delete a directory and its contents.

QUESTIONS

1. From your home directory, what is the output of `ls -R experiment*`?
2. What command did you use in step 12 to copy and rename the directory? You can use the `history` command to get a list of your most recently used commands. You can also use the up and down arrow keys to scroll through your previous commands.
3. Why did the `rmdir` command give an error in step 13 of the **Files and Directories** section?
4. What command did you eventually use to delete the directory?
5. Read the man pages for `ls`.
   a) What is the difference between `ls -R` and `ls -r`?
   b) What does `ls -a` do?
   c) What is special about the name of a hidden file?
   d) What is the name of the hidden file in the `~/experiments/exp1/data` directory?
6. Every directory contains hidden directories named `.` and `..`
   a) Where do they refer to?
   b) How can you list all of the files and directories, both hidden and visible, except . and ..?
7. How would you sort files
   a) by size?

b) by modification time?

8. How would you use `cd` to change from `~/experiments/exp1/data` to `~/experiments`
   a) using relative paths?
   b) using absolute paths?
9. Open `generate.sh` in `less` and `more`.
   a) How can you force `less` to display line numbers?
   b) What is the contents of line 21 of `generate.sh`
10. Read the man pages for `head`, `tail` and `wc`
    a) How many lines do `head` and `tail` show by default?
    b) How can you make `head` show 5 lines?
    c) Which argument makes `wc` show only the number of lines?

FILE PERMISSIONS

Linux is, in general, very good at looking after itself. There are times, however, when you'll need to perform some basic system administration.

One example of this is changing file permissions. Most Linux file systems have a simple but powerful set of file permissions that dictate who can do what with particular files and folders. Being able to mange them is extremely important – especially when writing software.

1. Use the `touch` command to make a new file called `important` in the `Lab01` folder.
2. Use the *long listing format* for `ls` to display the file permissions and other information, check the manual pages if you're unsure how to do this.

   Look for the line similar to the one below:

   ```
   -rw-r--r--  1 james james   0  2010-06-15 15:50 important
   ```

   We're most interested in the first part, `-rw-r—r--`. The file is readable by everyone but writeable only by its owner (lecture 2 contains information on reading file permissions).
3. Since the file is important you're going to protect it by making it read-only. This is done using the `chmod` command. Run `chmod a-w important`. This will modify the file permissions for all users (`a`) and remove (`-`) their write permissions (`w`).
4. Open the file in a text editor, add some text and try to save the file, it will not let you. Now try to delete the file using `rm important`. You will be asked to confirm the deletion, offering you a second chance before the file is deleted forever.

File permissions also determine whether or not a file is allowed to be executed (run). In the interest of security, files are set to be non-executable by default. You will need to learn how to make files executable before writing bash scripts in lab03.

1. In the `Lab01` directory there is a C program called `secret` containing the answer to one of the questions for this lab. If you to try run it you will get a “permission denied” error since it's not executable. Use `chmod` to make it executable using the `+x` flag.
2. Run it using `./secret` and note down the answer for later.

The `chmod` command and file permissions can be quite confusing. Read the manual pages if you have not already and get used to modifying file permissions.

QUESTIONS

11. There is a file with the following permissions `-rw-rw---x`

    a) Describe who can do what with the file.

b) Which command would change the permissions so that only the owner can write to and execute the file?

12. How can you tell, using only the file permissions, if a file is a directory?

13. Read the man pages for `rm`.

    a) How can you force `rm` to ask before deleting every file?

    b) How can you stop `rm` from asking before deleting a write protected file?