CS2106: Operating Systems
AY23/24 Semester 2
Introduction to Linux

Section 1. Introduction

If you used Unix or one of the Unix variants before, then most of the introductions below should be familiar to you. Feel free to skip all the parts which you already understood. For example, if you used SoC's unix server ***stu*** (by SSH into stu.comp.nus.edu.sg) before in other modules, then the differences are quite minor.

1.1 Getting access to Linux

Here's a couple of ways to get access to a Linux installation. Take note of **option** 1, as we will use it as the **standard test platform** for evaluating your lab submissions.

Option	Details
1.	**Use SoC Compute Cluster** **Requirements:** • You need SoC account to use these nodes. Please go to https://mysoc.nus.edu.sg/~newacct/ to (re)activate your account. • You need to enable **SoC Compute Cluster.** Please go to https://mysoc.nus.edu.sg/~myacct/services.cgi to enable the facility. **NOTE:** The clusters are using a different filesystem from your stu.comp.nus.edu.sg home directory. Thus you will not be able to see your files from your normal home directory. The SoC clusters use a load management system called "slurm". The slurm system allows you to log in to a common login node (see later), then use slurm commands to send work to the cluster nodes. The differences in using slurm versus just compiling and running on a server is quite minimal, requiring you to just add the "srun" command. **Usage:**

<table>
<tr><td></td><td>

Use any SSH Client to connect to xlog.comp.nus.edu.sg server using your SoC Credentials. Note: If you are connecting from outside NUS, you must connect to the SoC VPN, NOT the NUS VPN. Please see https://dochub.comp.nus.edu.sg/cf/guides/network/vpn for how to connect to the SoC VPN.
You will now be in one of the login nodes (xlog0 to xlog2). From here you can submit Slurm jobs:
<ol type="a">
Type “salloc” (without the quotes)to be given a job allocation. This allocates a node from the cluster for you to run your program. Slurm replies with:


162696 is a job allocation number. You will see a different number but that doesn’t matter.
To run your program on slurm, precede all commands with srun. The “srun” command acquires resources (e.g. nodes) to run your command and helps the cluster balance loads. If you don’t use srun you will end up doing everything on the login node xlog. To compile a program in C called test.c, without slurm we would do:
gcc test.c -o test
On slurm, we would do:
srun gcc test.c -o test
Aside from having to precede our commands with srun, running programs on slurm is pretty much the same as running on a normal Linux server.
Note that you can get a shell on one of the cluster nodes by doing srun --pty /bin/bash.

</td></tr>
<tr><td>2.</td><td>Install Ubuntu 20.04 natively on your PC

A great way to experience Linux on your own machine. You can gain useful experience "playing" with Ubuntu to learn the common operations on Linux</td></tr>
<tr><td>3.</td><td>Use VM on your PC to run Ubuntu 20.04

Instead of going for a full native installation. You can use VM tool, e.g. VirtualBox on your PC to run Ubuntu. You can get Ubuntu here: https://ubuntu.com/download</td></tr>
</table>

1.2 The Command Line Interface (CLI)

Whenever your login to the Compute Cluster Node (or start a terminal on Ubuntu).The terminal you started is actually composed of two programs. One is a graphical application which gives a window and a terminal emulation. The other is a **shell** which interprets ASCII command lines which is connected to the terminal. The **shell** uses a command line interface, because it is text based. You type in commands to ashell which performs the command(s) in that line of text. In Windows, there is also acommand line shell, **cmd** which has a similar purpose.

There are a number of different shells available under Linux. **BASH** (**B**ourne **A**gain**SH**ell) is the default and is a GNU enhancement descended from the original Unix shell, the **Bourne shell** (**sh**). You can write ***shell scripts*** which are small programs in the shell language to run a series of shell commands.

1.3 Using the built-in manual page (man)

(Note: For relatively small jobs like reading man pages, listing files in the directory or copying, it should be ok to run the commands directly on the xlog login node (xlog0, xlog1 or xlog2) instead of using srun to acquire resources to run the commands. However, for most tasks like compiling and running your lab programs, please use srun to allow slurm to balance the resources)

Most Unix commands and C built-in libraries have built-in documentation. Use the *Unix Manual* command: **man** to access these documentation:

man XXXX

where **XXXX** is either the command/C built-in library call/etc that you want to knowmore about. For example, the **man**command itself is self-documenting:

$ man man

The above means “get the manual page for the command **man**”Try

out:

$ man fprintf

Which explain the C-Library function **fprintf()**. Take note that Unix man pagesare written in a terse and very technical fashion. So it works more like a reference rather than a user-friendly tutorial. For a taste of a very long and scary looking manpage, try "**man gcc**"!

The manual pages are organized using sections, and an entry might be in more than one section, e.g. **man 1 login** documents the command line login program while **man 3 login** documents the files which record the users of the system. You can search for keywords with the **-k** option, eg. **man -k login**.

The various manual sections have their own introduction. Try **man intro** or**man 1 intro**. System calls are described in **man 2 intro**. The **man** uses a **pager**, which is another program to display one page at a time on a terminal. You can pageforward with **[SPACE]**, backward with '**b**' and quit the manual page at any time with'**q**'.

1.3 Listing Files and Directories

The **ls** command will list all the files in the specified directories. You can use the **-l** option for a long listing format which displays information like size, permissions, owner, group, creation date, etc. If you do not specify a directory, by default it usesthe current directory. The **-R** option causes **ls** to list recursively.

Some examples to try:
$ **ls**
$ **ls -l**
$ **ls /**
$ **ls -l /**

1.4 Editing Files

You can use various text editors either directly from the desktop or the terminal.Some editors are strictly text based and some have a graphical interface in X. Manyeditors are available such as: **emacs, vi, vim, gedit**, etc. Editors like **emacs** and **vi** / **vim** are not so easy to use but are fairly powerful and popular with programmers. You are encouraged to learn **vi**/**vim** or **emacs,** many tutorials are available on the web. (Note: **vim** and **emacs** are available on Windows too).

If you are new to **Unix**, you can use **gedit** / **nano** which is more user friendly. On the slurm servers you should invoke your editor using srun because you are likely to be running it for a while. To do so with vim, for example:

srun --pty vim myfile.txt

The --pty option tells srun to give you an interactive session so that you can work with vim. Without this option vim will be running on a node but you will not be able to interact with it.

1.5 Backup and Transferring Files

If you need to transfer files and folders to / from the compute cluster node, thereare a few common choices:

1.5.1 scp: Secure Copy. A command you can type to move files between Compute Cluster Node and stu. E.g. to transfer a file called "lab1codes.zip" to your cluster account:

E.g.: `scp lab1codes.zip <username>@xlog.comp.nus.edu.sg:~/`

This will copy lab1codes.zip to your home directory on the cluster by coping to xlog (<username> is your xlog user ID, which is the same as the user ID you use to log in to stu.comp.nus.edu.sg). Note that you **SHOULD NOT** copy the files to your stu.comp.nus.edu.sg home directory as files there will not appear in the clusters filesystem.

1.5.2 git: Popular version control software. ** Please ensure your git repository for CS2106 labs is **private.** **

1.6 Some Things to Remember!

The following are notable points for using Unix in general and also applicable toLinux:

1.6.1 Unix is *case sensitive*. Most commands are lowercase.

1.6.2 Unlike Windows, Unix **has no drive letters** (i.e. no C:, D:, E: etc). Everything isin some directory (i.e. folder).

1.6.3 Unix uses forward slash (**/**) to separate directory names, while Windows uses backslash (****).

1.6.4 The ***** wildcard is treated uniformly by Unix shells. In Windows, ***** worksdifferently for different programs.

1.6.5 It is worthwhile to look first at the **man** page of a command

1.6.6 Try using various command line programs. Some programs need administrator privileges (in Unix, this is the root user who has *superuser privileges*) to run.

Section 2 Sample Usage Session

Skip this section if you already know how to use Unix / Linux.

1. **mkdir junk** // makes a new directory (i.e. folder) ***junk***
2. **cd junk** //change directory to it, you are now in ***junk***
3. **pwd** //prints the path of your current directory
4. **echo abcd > test1.txt** // makes a new file **test1.txt** with contents"**abcd**"
5. **less test1.txt** //display **test1.txt** using the pager. `**q**' will quitfrom less and `**h**' will give the help screen.

Select an editor to use (see previous editing section).

1. edit the file **test1.txt**, eg.

   srun --pty nano test1.txt //or any editor of your choice. Here we use
   // slurm as we may potentially use nano for
   // a long time. Note the --pty option that creates
   // a shell for you to interact with nano.

   OR

   nano test1.txt // On your own Ubuntu or other Linux box.

2. change the content of the file (currently "**abcd**") to something else
3. save the file in the editor (this is editor specific)
4. **cat test1.txt** //concatenate 1 or more files, and print tooutput, so this displays **test1.txt** because output is the terminal
5. **cat test1.txt test1.txt > test2.txt** //**test1.txt** is concatenated twice and the output saved to **test2.txt**
6. **cp test1.txt test3.txt** //copies **test1.txt** to **test3.txt**
7. **ls** //listing should show 3 fiels
8. **rm test1.txt** //deletes **test1.txt**
9. **ls** //only **test2.txt** and **test3.txt left**
10. **rmdir ../junk** //try to remove directory, complains about non-empty directory
11. **rm test*.txt; cd .. ; rmdir junk** //no more files indirectory, go up to parent directory, remove junk directory. You can perform multiple commands in a single line by using the ";" separator.

12. **logout** // or you can just press <Ctrl-D>