CS2106: Introduction to Operating Systems
AY19/20 Semester 2
Introduction to OS labs

This document serves as a guide for you to start working with our Linux servers. If you are familiar with Unix or any of its variants (such as Linux), you may skip the sections that you are familiar with.

For example, if you have used SoC's Unix server ***sunfire*** (by SSH into `sunfire.comp.nus.edu.sg`) before in other modules, you should be already familiar with most of the commands. Please note that **Solaris** (a Unix variant by Oracle) is used on the sunfire server while **Linux** is used on the rest of the machines.

For CS2106 labs, you will be using Linux machines in the SoC cluster:
https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/hardware

While you can use any of the available machines, we suggest that you use machines xcnd10 to xcnd19, which are dedicated to the CS2106 labs.

1 Accessing the Linux Servers

1. 1 Logging In

For OS labs, you will be using one of the servers with the following hostnames:

```
xcnd10.comp.nus.edu.sg
xcnd11.comp.nus.edu.sg
xcnd12.comp.nus.edu.sg
   …
xcnd19.comp.nus.edu.sg
```

You will be **remotely connecting to the servers over Secure Shell (SSH)**. To connect over secure shell, you need an SSH client utility. Depending on your computer's operating system, please use one of the following methods.

Linux, Mac, Windows 10 1803 or newer:

Users of Linux, Mac and Windows 10 1803 or newer may use the in-built SSH client through the console (CLI). Type in the following command and press enter. Once prompted, please enter your password.

```
ssh <username>@xcnd10.comp.nus.edu.sg
```

Windows 10 1709 and older:

There is no in-built SSH client on this version of Windows. You need to download one of the freely available SSH client software such as Putty (https://www.putty.org/) and install it on your computer.

Enabling Cluster Access:
For security reasons, you may first need to give yourself access to the cluster. The instructions for doing so can be found here:
https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/enable-disable-access

Outside of SoC network:

If you are accessing the servers from outside of SoC network, you may need to first connect to `sunfire` server over SSH and then connect to cluster over SSH from sunfire. Note that accessing the `sunfire` server requires an SoC account.

Alternatively, you may want to use the SoC VPN to directly access the servers over SSH from anywhere. To do so, download the FortiClient tool, set up a VPN connection with the remote gateway pointing to `webvpn.comp.nus.edu.sg` and log in with your NUSNET credentials. No SoC account required. Note that due to occasional errors in the NUS network, sometimes you cannot access the cluster directly from SoC (so you have to first log in to `sunfire`). However, you can always access the cluster directly via the SoC VPN.

1.2 Your home folder

The SoC students already have their home folders on the servers in the cluster:
The location of the home folder is: `/home/j/<username>`
E.g.: `/home/j/sunimal`

Your home folder is not shared with other students or staff. You may keep your working files related to CS2106 labs in the folder and they will not be deleted upon logging out. Your files are accessible from any machine in the cluster, so you don't have to stick to the same machine every time. However, please back up your files regularly to your own computer.

1.3 Students without an SoC Unix account:

If you are not an SoC student, you may not have an SoC Unix account yet. We have created a temporary user account for you on the `rachmaninoff3` server. The username is your NUSNET id in the form of ***E0123456*** (upper case). The password is your student number

in the form of ***A0123456Z*** (upper case). Please **change your password immediately upon the first login.**

To connect to the server, type in your terminal the following:

```
ssh <username>@rachmaninoff3.d2.comp.nus.edu.sg
```

Once logged in, you must change the initial password to your own personal password. Choose a password which you can remember. PLEASE DO NOT LOSE THE PASSWORD. Use the **`passwd`** command to change your password. An example usage is given below:

```
[user1@rachmaninoff3 ~]$ passwd
Changing password for user user1.
Changing password for user1.
(current) UNIX password: <current password>
New password: <new password>
Retype new password: <new password>
passwd: all authentication tokens updated successfully.
```

The location of your home folder is: `/cs2106/<username>`
E.g.: `/cs2106/E0123456`

Note that your home folder on `rachmaninoff3` is not in synch with your future home folder on the machines in the cluster. Once your SoC account is created, you should switch to the cluster machines and copy your files over there.

2 The Linux Command Line Interface (CLI)

2.1 Linux Shell

Once you are logged in to the server over SSH, you are connected to the Linux shell (Command Line Interface) of one of our servers. You type in commands to a shell which performs the command(s) in that line of text. In Windows, there is also a command 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**). A list of shells of typical shells in Unix are: **`sh, bash, csh, ksh, tcsh, zsh`**. (Not all these shells may be available). You can stick to the default choice of **`bash`** unless you feel like trying out other shell(s). For interactive usage, **`bash`** and **`tcsh`** are the popular shells. Shells are also used for writing shell scripts (this may be a non-interactive use of the shell) which are small programs in the

shell language and some of the other shells can be convenient/efficient for that purpose. If you want to change you default shell, you can use the **chsh** program.

2.2 Using the built-in manual page (man)

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

```
man XYZ
```

where **XYZ** is either the command/C built-in library call/etc that you want to know more about.
For example, it 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().** Other commands which you may have tried are also in the manual page, e.g., **man date**. Take note that Unix man pages are 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 man page, 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 page forward with **[SPACE]**, backward with '**b**' and quit the manual page at any time with '**q**'.

2.3 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 uses the current directory. The **`-R`** option causes **`ls`** to list recursively.

Some examples to try:

```
$ ls
$ ls -l
$ ls /
$ ls -la /
```

2.3 Editing Files

There is a multitude of CLI based text editors available for Linux such as vim, nano, emacs etc. We recommend using vim editor because it is a widely used, powerful and customizable text editor. It may take some time to get used to, but once you get used to it, you will want to use if for everything. :-)

Starting Vim

To edit an existing file or start a new file, type `vim filename` (e.g., `vim myfile.c`).

Using Vim

There are two modes, Command Mode and Insert Mode.
(1) Command mode is used for moving around the file quickly and to make use of the huge number of commands that make vim such a powerful text editor.
(2) Insert Mode is used just for typing in text.

Inserting Text

When you open a file using vim, by default you are in the Command Mode. To switch to Insert Mode, press `i` or `a`. “INSERT ” will then appear at the left bottom of the screen.

Pressing `i` (for insert) will put the cursor before the currently highlighted position, while pressing `a` (for append) will put the cursor after the currently highlighted position.
To return from Insert Mode back to Command Mode, just press the ESC key on the top left corner of your keyboard.

Moving around in your program

There are two options here:
(1) Use the **arrow keys** on your keyboard
(2) Use the `hjkl` keys. (`h` - left, `j` - down, `k` - up, `l` - right)

Undo and Redo
`u` - Undo your last action
`ctrl + r` - Redo your last undo

Save your work
`:w` - Save the file you are working on.
`:wq` - Save your work and exit vim.
`:q!` - Quit vim without saving your modification.

Copy and paste
`yy` - Yank (copy) 1 line
`yy` - Copy lines (e.g., 5yy will copy the next 5 lines since the current line)
`p` - Paste after cursor
`P` - Paste before cursor

Running Unix commands from inside vim

```
:!gcc -Wall %
:!a.out
```

The above sequence of commands will compile and run your program from within vim. Remember to save the file first or the previous saved copy will get compiled.

There is much more about vim and you can refer to one of the thousands of vim guides online such as this (https://vim.rtorr.com/) to learn more useful vim commands.

2.4 Logging Out

When you are done, please log out from Linux.

```
[user1@rachmaninoff3 ~]$ logout
```

2.5 Backup and Transferring Files

You can transfer the files out of the OS lab server over SCP (ssh file transfer). Following example shows how to transfer myfile.c to sunfire server.

```
scp myfile.c user1@sunfire.comp.nus.edu.sg:.
```

You may also install an SCP client such as FileZilla (https://filezilla-project.org/) on your computer to easily transfer files between your computer and a remote server.

2.6 Some Things to Remember!

The following points are to be noted for Unix in general and also Linux:

- Unix is *case sensitive*. Most commands are lowercase.
- Unlike Windows, Unix **has no drive letters** (i.e. no C:, D:, E: etc). Everything is in some directory.

- Unix uses forward slash (**/**) to separate directory names, while Windows uses backslash (****).
- The ***** wildcard is treated uniformly by Unix shells. In Windows, ***** works differently for different programs.
- It is worthwhile to look first at the man page of a command
- Try using various command line programs. Some programs need administrator privileges (in Unix, this is the root user who has *superuser privileges*) to run.

3 Sample Usage Session

You may skip this section if you already know how to use Unix / Linux.

1. **`mkdir junk`** // makes a new directory 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 quit from less and `**h**' will give the help screen.

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

1. edit the file **`test1.txt`**, eg.
   `vim test1.txt` //or any editor of your choice
2. change the content of the file "**`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 to output, 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 shows the 3 _les
8. **`rm test1.txt`** //deletes **`test1.txt`**
9. **`ls`** //only **`test2.txt`** and **`test3.txt`**
10. **`rmdir ../junk`** //try to remove directory, complains about non-empty directory
11. **`rm test*.txt; cd .. ; rmdir junk`** //no more _les in directory, go up to parent directory, remove junk directory. You can perform multiple commands in a single line by using the ";" separator.
12. **`logout`**

Updated 3 Feb 2020
Some sections partially adopted from the VIM guide by Dr. Zhou Lifeng, SoC, NUS.