

## **Lab 1: Virtual Machine Setup, Introduction to C programming, Github and Linux (5 Marks)**

Student Name: Muhammad 'Asif Danial Bin Mohd Shahir

Github Repository for Lab 1: <https://github.com/asfdnl/ITT440-Lab1>

Matric No: 2021114813

Group: CS2554A

### **1.1 Linux Virtual Machine Setup**

Virtual machine is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Ubuntu is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: Desktop, Server, and Core. All the editions can run on the computer alone, or in a virtual machine. You will be using Ubuntu server for this lab.

#### **i. Installing Virtual Box**

Oracle VirtualBox is a hypervisor which allows you to emulate an operating system on your own PC and use it like it's running on real hardware. The emulated host running on hypervisor is called as virtual machine.

VirtualBox is Free but Proprietary software by Oracle Corporation; you can download it from here <https://www.virtualbox.org/wiki/Downloads>

After finish downloading, install and run the hypervisor on your machine.

#### **ii. Install Ubuntu Virtual Machine**

You can install Ubuntu 18.04.3 LTS from either bootable image (.iso) or virtual machine image (.ova). You can download both types from here <https://ubuntu.com/download/server>

If you install via bootable image (.iso), create new virtual machine in your hypervisor and boot the image. Follow the installation instruction.

If you install via VM image (.ova), you can import it into the hypervisor, it may takes a while. After finish with installation please run you kali Ubuntu virtual machine

#### **iii. Make sure VM has internet connectivity**

Please make sure you configure your virtual machine network adapter as NAT (network address translation). You can test internet connectivity by opening a terminal on Ubuntu and try to ping google or cloud flare DNS. **ping 8.8.8.8 / ping 1.1.1.1**

#### iv. Run and Explore Ubuntu Linux

After finish with installation, it's time to explore ubuntu and software/tools that came with it, please answer this question

Please provide the output and screenshot when execute “**uname -a**” in terminal.  
Please explain the output **(0.5 Marks)**

```
asfdnl@dannyphantom:~$ uname -a
Linux dannyphantom 5.4.0-88-generic #99-Ubuntu SMP Thu Sep 23 17:29:00 UTC 2021 x86_64 x86_64 x86_64
GNU/Linux
asfdnl@dannyphantom:~$
```

The output above show the user that can access the Ubuntu machine.

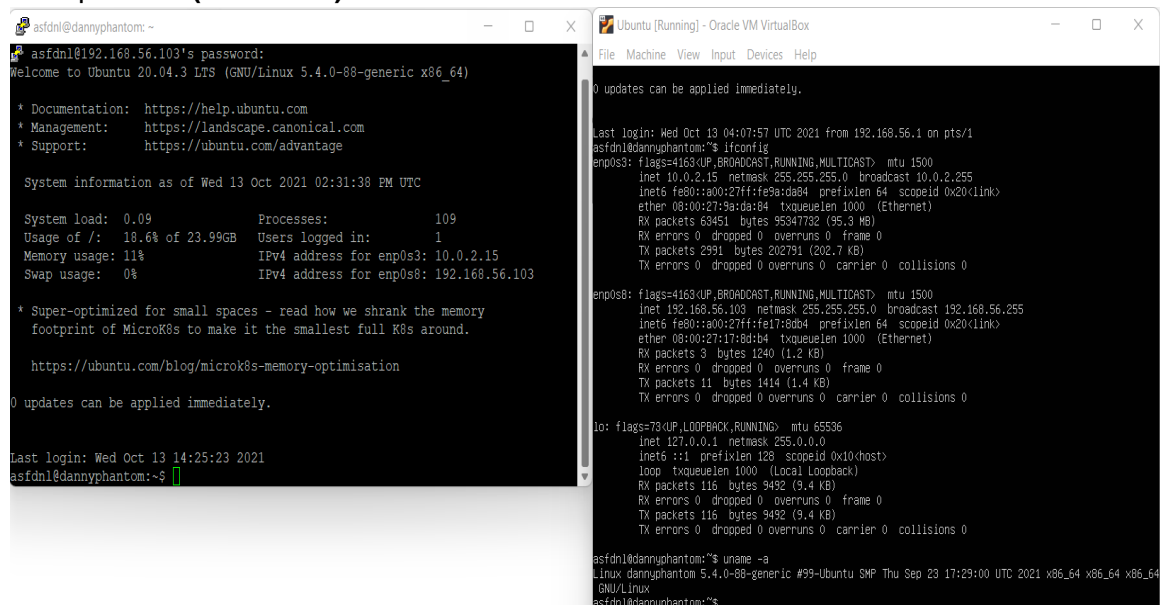
#### v. Access your Ubuntu Linux remotely via SSH protocol

Network administrator usually access their linux sever via SSH protocol. Before you can access your linux via SSH, you need to install SSH server on your Ubuntu virtual machine. You can follow this tutorial <https://linuxize.com/post/how-to-enable-ssh-on-ubuntu-18-04/>

After you have enabled SSH on your Ubuntu machine, you need to install SSH client on your window machine. You can download ssh client from here <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Once you have installed ssh client on your window machine, you can ssh to your Ubuntu using putty by using your Ubuntu IP address and username. You can check your Ubuntu machine IP address by using this command on Ubuntu terminal **#ifconfig**.

Please provide screenshot that you able to access Ubuntu machine on Putty using SSH protocol **(0.5 Marks)**



## 1.2 Introduction to Version Control and GIT

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. You will be using git for Lab work and project through this course

### i. Installing GIT and exploring GIT

You can use this tutorial to install git in Linux  
<https://www.atlassian.com/git/tutorials/install-git>

This is a complete tutorial in using GIT  
<https://www.atlassian.com/git/tutorials>

Please provide screenshot that you have install GIT in your Linux **(0.5 Marks)**

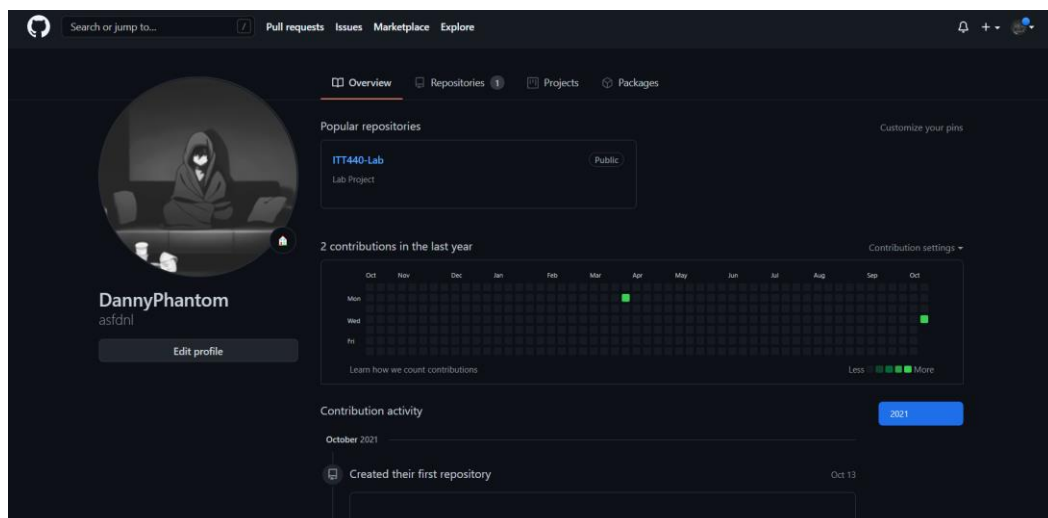
```
asfdnl@dannyphantom:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3.2).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

asfdnl@dannyphantom:~$ git --version
git version 2.25.1
asfdnl@dannyphantom:~$
```

### ii. Signing up your Github Account

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. You'll create your own Hello World repository and learn GitHub's Pull Request workflow, a popular way to create and review code.

Please sign-up a git account (<https://github.com/>) and provide screenshot of your github profile **(0.5 Marks)**



### 1.3 Introduction to C and Linux

For the lab you are required to have a distribution of Linux in order for you to compile the program as. Please create a github repository for this lab exercise and push your code to your github repository

well as checked whether your program have run correctly.

After this lab, students should be able to:

- Create, Edit, save a file using Linux
- Compile a C program
- Check for open port and running program

#### i. Hello World Program in C

1. Open up a terminal.

2. Create a new file by using the following command:

`touch helloworld.c`

3. Edit the file by using nano text editor.

`nano helloworld.c`

4. Type in the following program segment:

```
#include <stdio.h>
int main(void) {
/* This is my first program in C */
printf("Hello World!");
printf("I Love C");
return (0);
}
```

5. Save the program by pressing ctrl O. When prompted for file name, press ENTER.

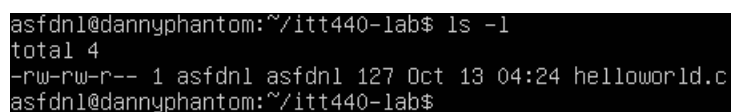
6. Exit from nano text editor by pressing ctrl X

7. The file should have been saved inside your home directory, to verify run the following command:

`ls -l`

You should see the file listed along with other files inside the directory.

Please provide screenshot of file inside the directory **(0.5 Marks)**



```
asfdnl@dannyphantom:~/itt440-lab$ ls -l
total 4
-rw-rw-r-- 1 asfdnl asfdnl 127 Oct 13 04:24 helloworld.c
asfdnl@dannyphantom:~/itt440-lab$
```

## ii. **Compiling a C Program**

In linux, the popular compiler is known as GNU C Compiler or gcc in short. For this course, we are going

to use gcc to compile our C program.

1. Open up a terminal.

2. Go to your home directory or directory where you have saved your helloworld.c. To go to your

home directory type `cd` or `cd /home/your_username`.

3. Compile the program by using the following command:

`gcc helloworld.c`

4. If you get an error, edit the program again using nano and recompile. If you get no error (after

running the above command, you are returned to the prompt) then congratulations you have

successfully compile your first C program.

5. A succesful compilation will create an object file called a.out. Check whether the file exist by

running:

`ls -l`

You should see a file called a.out

6. To see the output of the program, you can run the file by running the following command:

`./a.out`

Please provide screenshot of the program output **(0.5 Marks)**

```
asfdn1@dannyphantom:~/itt440-lab$ gcc helloworld.c
asfdn1@dannyphantom:~/itt440-lab$ ls
a.out  helloworld.c
asfdn1@dannyphantom:~/itt440-lab$ ./a.out
Hello World
I love C
asfdn1@dannyphantom:~/itt440-lab$ _
```

### iii. Assigning Names to the output file

One problem with having the output file name as a.out is that once we compile a new program, we are unable to run our previously compiled program without recompiling the program. In order to overcome

this program, we could give a name to the output file during compilation.

1. Open up a terminal.

2. Go to your home directory or directory where you have saved your helloworld.c. To go to your home directory type `cd` or `cd /home/your_username`.

3. Compile the program by using the following command:

```
gcc -o helloworld.out helloworld.c
```

the option `-o` let us specify the name of the output file.

4. A succesful compilation will create an object file called helloworld.out. Check whether the file

exist by running:

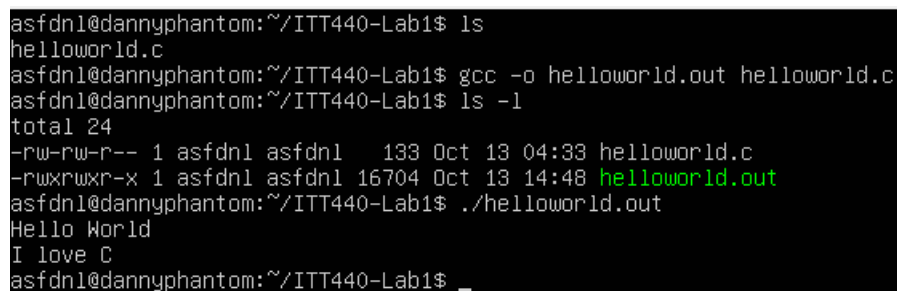
```
ls -l
```

You should see a file called helloworld.out instead of a.out.

5. To see the output of the program, you can run the file by running the following command:

```
./helloword.out
```

Please provide screenshot of the program output **(0.5 Marks)**



```
asfdn1@dannyphantom:~/ITT440-Lab1$ ls
helloworld.c
asfdn1@dannyphantom:~/ITT440-Lab1$ gcc -o helloworld.out helloworld.c
asfdn1@dannyphantom:~/ITT440-Lab1$ ls -l
total 24
-rw-rw-r-- 1 asfdn1 asfdn1 133 Oct 13 04:33 helloworld.c
-rwxrwxr-x 1 asfdn1 asfdn1 16704 Oct 13 14:48 helloworld.out
asfdn1@dannyphantom:~/ITT440-Lab1$ ./helloworld.out
Hello World
I love C
asfdn1@dannyphantom:~/ITT440-Lab1$ _
```

#### iv. Getting a variable from user

1. Open up a terminal.

2. Create a file called userinput.c by using nano text editor.

nano userinput.c

3. Type in the following program segment:

```
#include <stdio.h>
int main(void) {
/* This is my second program in C */
int age;
printf("Hi, how old are you? > ");
scanf("%d", &age);
printf("You are %d years old", age);
return (0);
}
```

4. Save the program by pressing ctrl O. When prompted for file name, press ENTER.

5. Exit from nano text editor by pressing ctrl X

6. The file should have been saved inside your home directory, to verify run the following command:

ls -l

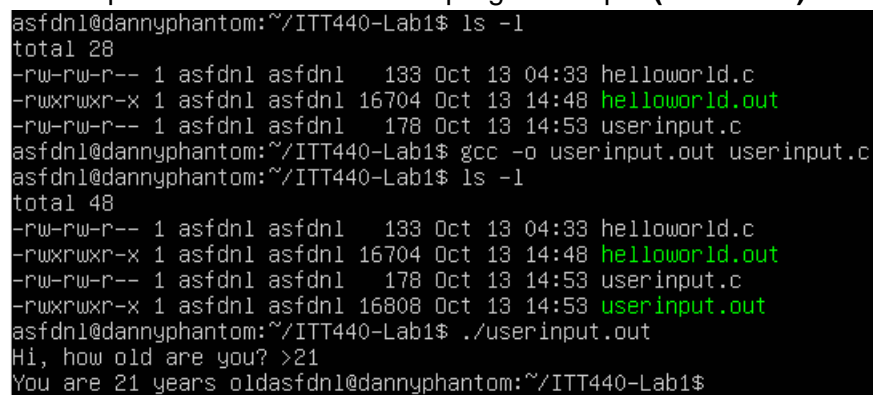
You should see the file listed along with other files inside the directory.

7. Compile the program by using the following command:

gcc -o userinput.out userinput.c

8. If you get an error, edit the program again using nano and recompile.

Please provide screenshot of the program output **(0.5 Marks)**



```
asfdnl@dannyphantom:~/ITT440-Lab1$ ls -l
total 28
-rw-rw-r-- 1 asfdnl asfdnl 133 Oct 13 04:33 helloworld.c
-rwxrwxr-x 1 asfdnl asfdnl 16704 Oct 13 14:48 helloworld.out
-rw-rw-r-- 1 asfdnl asfdnl 178 Oct 13 14:53 userinput.c
asfdnl@dannyphantom:~/ITT440-Lab1$ gcc -o userinput.out userinput.c
asfdnl@dannyphantom:~/ITT440-Lab1$ ls -l
total 48
-rw-rw-r-- 1 asfdnl asfdnl 133 Oct 13 04:33 helloworld.c
-rwxrwxr-x 1 asfdnl asfdnl 16704 Oct 13 14:48 helloworld.out
-rw-rw-r-- 1 asfdnl asfdnl 178 Oct 13 14:53 userinput.c
-rwxrwxr-x 1 asfdnl asfdnl 16808 Oct 13 14:53 userinput.out
asfdnl@dannyphantom:~/ITT440-Lab1$ ./userinput.out
Hi, how old are you? >21
You are 21 years oldasfdnl@dannyphantom:~/ITT440-Lab1$
```

**v. Checking for open port**

There are times when we are required to check list of port open at our PC, to do this we can utilize the

*netstat* command:

1. Open up a terminal.

2. Run the following command:

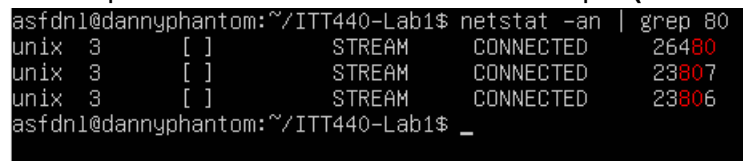
`netstat -an | grep <port no.>`

For example if you are interested in checking for port 80, you would run the command as:

`netstat -an | grep :80`

3. If there is a result, it shows you that there are currently port 80 opened in your computer ( or you are accessing port 80 at the remote location.

Please provide screenshot of the netstat output **(0.5 Marks)**



```
asfdn1@dannyphantom:~/ITT440-Lab1$ netstat -an | grep 80
tcp 0 0 0.0.0.0:80 LISTENING 0
tcp 0 0 0.0.0.0:80 LISTENING 0
tcp 0 0 0.0.0.0:80 LISTENING 0
asfdn1@dannyphantom:~/ITT440-Lab1$ _
```

**vi. Checking for a running program**

At times, we wanted to know if a program is running (for example a child program) then we can use the

*ps* command to check.

1. Open up a terminal

2. Run the following command:

`ps aux | grep <command>`

replace command with the name of the program that you are interested in.

3. If you get a result, it shows that the program is still running

Please provide screenshot of the ps output **(0.5 Marks)**



```

root      580  0.0  0.0    0    0 ?      S<   14:17  0:00 [loop4]
root      581  0.0  0.0    0    0 ?      S<   14:17  0:00 [loop5]
root      592  0.0  0.0    0    0 ?      S    14:17  0:00 [jbd2/sda2-8]
root      593  0.0  0.0    0    0 ?      I<   14:17  0:00 [ext4-rsv-conver]
systemd+  607  0.0  0.3  90228  6140 ?    Ss1  14:17  0:00 /lib/systemd/systemd-timesyncd
systemd+  646  0.0  0.3  26732  7816 ?    Ss   14:17  0:00 /lib/systemd/systemd-networkd
systemd+  648  0.0  0.6  23896 11980 ?    Ss   14:17  0:00 /lib/systemd/systemd-resolved
root      661  0.0  0.4  239296  9368 ?    Ss1  14:18  0:00 /usr/lib/accounts-service/accounts-
-daemon
root      664  0.0  0.1   6812  2788 ?    Ss   14:18  0:00 /usr/sbin/cron -f
message+  665  0.0  0.2   7664  4680 ?    Ss   14:18  0:00 /usr/bin/dbus-daemon --system --a
ddress=systemd: --nofork --nopid
root      675  0.0  0.9  29076 18120 ?    Ss   14:18  0:00 /usr/bin/python3 /usr/bin/network
d-dispatcher --run-startup-trigg
syslog    676  0.0  0.2  224348  4760 ?    Ss1  14:18  0:00 /usr/sbin/rsyslogd -n -iNONE
root      678  0.6  1.7  639460 35672 ?    Ss1  14:18  0:00 /usr/lib/snapd/snapd
root      685  0.0  0.3   16652  7792 ?    Ss   14:18  0:00 /lib/systemd/systemd-logind
root      687  0.0  0.6  394952 13824 ?    Ss1  14:18  0:00 /usr/lib/udisks2/udisksd
daemon    690  0.0  0.1   3792  2232 ?    Ss   14:18  0:00 /usr/sbin/atd -f
root      694  0.0  0.0    0    0 ?      I    14:18  0:00 [kworker/0:5-events]
root      696  0.0  0.0    0    0 ?      I    14:18  0:00 [kworker/0:6-events]
root      699  0.1  0.0    0    0 ?      I    14:18  0:00 [kworker/0:7-events]
root      700  0.0  0.0    0    0 ?      I    14:18  0:00 [kworker/0:8-events]
root      703  0.0  0.1   6000  3972 tty1    Ss   14:18  0:00 /bin/login -p --
root      726  0.0  0.3   12176  6968 ?    Ss   14:18  0:00 sshd: /usr/sbin/sshd -D [listener
] 0 of 10-100 startups
root      727  0.0  1.0 107908 20956 ?    Ss1  14:18  0:00 /usr/bin/python3 /usr/share/unatt
ended-upgrades/unattended-upgrad
root      747  0.0  0.4  236440  9104 ?    Ss1  14:18  0:00 /usr/lib/policykit-1/polkitd --no
-debug
asfdnl    1015  0.1  0.4   18496  9604 ?    Ss   14:19  0:00 /lib/systemd/systemd --user
asfdnl    1017  0.0  0.1 103164  3304 ?    S    14:19  0:00 (sd-pam)
asfdnl    1022  0.0  0.2   8264  5188 tty1    S    14:19  0:00 -bash
asfdnl    1043  0.0  0.1   8892  3216 tty1    R+   14:20  0:00 ps waux
asfdnl@dannyphantom:~$ ps waux | grep 1043
asfdnl    1045  0.0  0.0   6300   672 tty1    S+   14:20  0:00 grep --color=auto 1043
asfdnl@dannyphantom:~$ _

```