Department of Computer Science and Engineering

The Chinese University of Hong Kong

CSCI/CENG 3150: Introduction to Operating Systems

Lab 01: Linux Environment and C Programming – Compile/Run

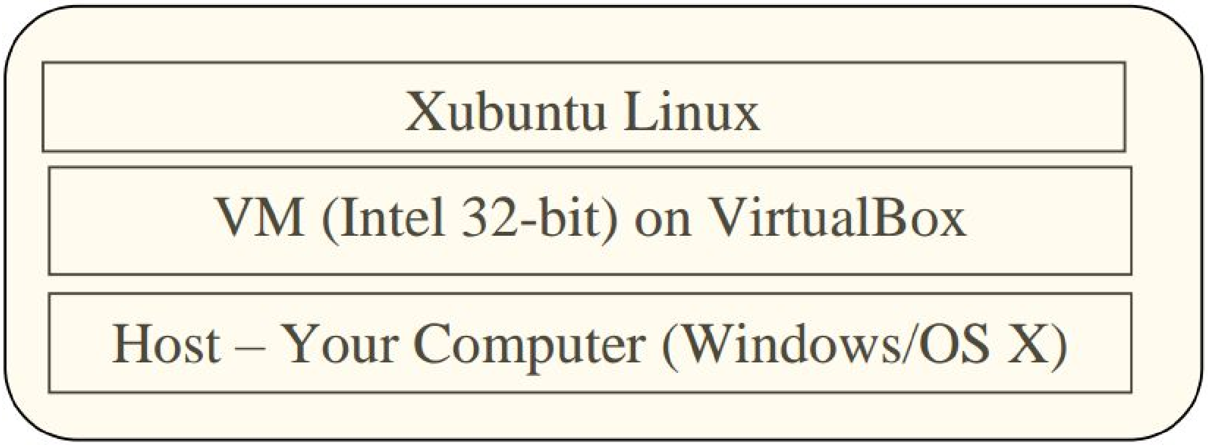
Objectives:

1. **Install the Linux environment on your computer.**
2. **C programming review – Compile/Run.**
3. **Makefile tutorial.**

Exercise (Deadline: 2019-09-18 23:59:59):

After this lab, you need to finish a small exercise in section 4 and submit it under the instructions in section 4 before deadline.

1. Linux environment installation (please finish this at home and ask TA if you have questions)


Xubuntu Linux
VM (Intel 32-bit) on VirtualBox
Host – Your Computer (Windows/OS X)

As shown above, we will use a Linux environment (Xubuntu) built on a virtual machine (VM). So in our computer, we need to first install VirtualBox which is a VM monitor, and then import a VM image that contains Xubuntu Linux.

The XUbuntu VM image can also be downloaded from:
https://www.cse.cuhk.edu.hk/~shao/zili_files/csci3150/3150_XUbuntu.ova)

The XUbuntu we use this year:

OS:	XUbuntu 18.04LTS (32 bit)
CPU:	4
Memory:	1GB
Disk:	10GB

gcc: 7.4.0
Username: csci3150
Password: csci3150

Please follow the link below to build up the Linux environment:
https://lumian2015.github.io/linuxBasic/

All assignments are programming-based, and will be graded based on the above Linux environment. Thus, it is extremely important for you to build up this environment in your computer.

2. C programming review - Compile/Run

Edit files: To edit a file, you can either (1) use text editor installed in vm (Sublime Text/vi/nano) (2) use other tools in your host machine and then transfer it via the shared directory.

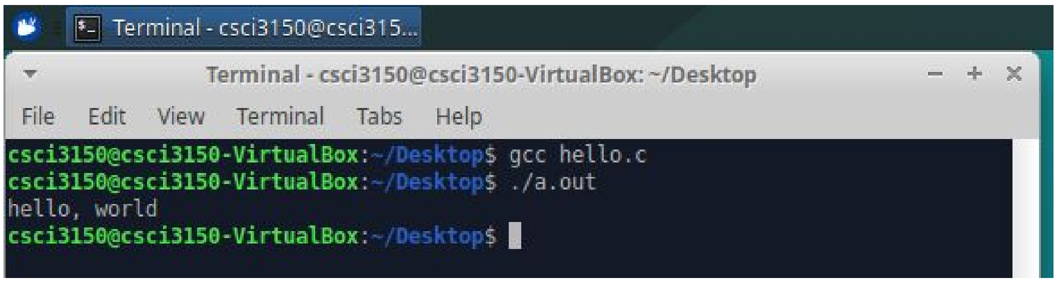
Now edit a file called **hello.c** with the following contents:

```
/* header files go up here */
/* note that C comments are enclosed within a slash and a star, and may wrap over lines */
// if you use gcc, two slashes will work too (and may be preferred)
#include <stdio.h>
/* main returns an integer */
int main(int argc, char *argv[])
{
        /* printf is our output function; by default, writes to standard out */
        /* printf returns an integer, but we ignore that */
        printf("hello, world\n");
        /* return 0 to indicate all went well */
        return(0);
}
```

Compile/Run: A C program can be compiled and run as follows:

```
prompt> gcc hello.c
prompt> ./a.out
```

(e.g. The screenshot from my computer shows as follows)


Terminal - csci3150@csci315...
Terminal - csci3150@csci3150-VirtualBox: ~/Desktop
File Edit View Terminal Tabs Help

```
csci3150@csci3150-VirtualBox:~/Desktop$ gcc hello.c
csci3150@csci3150-VirtualBox:~/Desktop$ ./a.out
hello, world
csci3150@csci3150-VirtualBox:~/Desktop$
```

Here are some useful compilation flags for gcc:

```
prompt> gcc -o hw hello.c            # -o: to specify the executable name
prompt> gcc -Wall hello.c            # -Wall: gives much better warnings
prompt> gcc -g hello.c               # -g: to enable debugging with gdb
prompt> gcc -O hello.c               # -O: to turn on optimization
prompt> gcc –o hw -g -Wall hello.c # Combine these flags
```

3. Makefile tutorial

For most of the assignments of this course, you will need to use makefile to compile them. As a result, we will go through a simple makefile tutorial.

In the following, there are 3 files which you can find them in folder ***example***.

hellomake.c

```c
#include <hellomake.h>

int main() {
    // call a function in another file
    myPrintHelloMake();
    return(0);
}
```

hellofunc.c

```c
#include <stdio.h>
#include <hellomake.h>

void myPrintHelloMake(void) {
    printf("Hello makefiles!\n");
    return;
}
```

hellomake.h

```c
void myPrintHelloMake(void);
```

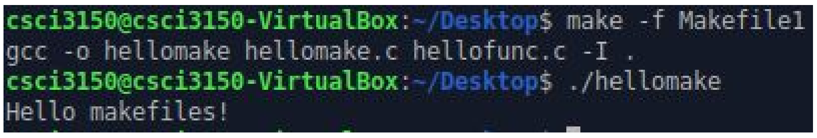
To compile them, the command you need is

```
gcc -o hellomake hellomake.c hellofunc.c -I .
```

Makefile1

```
hellomake: hellomake.c hellofunc.c
        gcc -o hellomake hellomake.c hellofunc.c -I .
```

Compile with Makefile1



```
csci3150@csci3150-VirtualBox:~/Desktop$ make -f Makefile1
gcc -o hellomake hellomake.c hellofunc.c -I .
csci3150@csci3150-VirtualBox:~/Desktop$ ./hellomake
Hello makefiles!
```

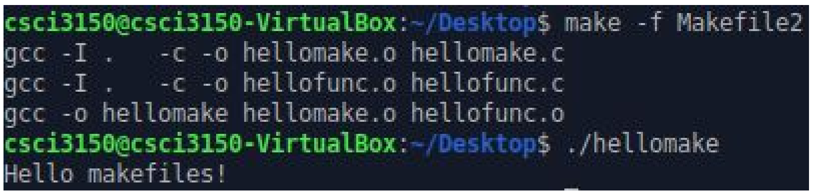
Makefile2

```
CC=gcc
CFLAGS=-I .

hellomake: hellomake.o hellofunc.o
        $(CC) -o hellomake hellomake.o hellofunc.o

clean:
        rm hellomake
```

Compile with Makefile2



```
csci3150@csci3150-VirtualBox:~/Desktop$ make -f Makefile2
gcc -I .   -c -o hellomake.o hellomake.c
gcc -I .   -c -o hellofunc.o hellofunc.c
gcc -o hellomake hellomake.o hellofunc.o
csci3150@csci3150-VirtualBox:~/Desktop$ ./hellomake
Hello makefiles!
```

4. Exercise (Deadline: 2019-09-18 23:59:59)

In the folder exercise, you can find a file ***main.c*** and two sub-folders. The ***main*** function in ***main.c*** will call the functions in those two sub-folders. You need to check the content and write a makefile which can compile them to an executable file called ***lab1***(note: we will compile with the makefile under the folder ***exercise***).

Please submit your makefile with the name **Makefile** to blackboard before the deadline.