



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

CSCI3150 - Introduction to Operating Systems

WANG Zhiqi
zqwang@cse.cuhk.edu.hk



Linux Environment Installation

We provide you an [image](#) with the following configurations:

- OS: XUbuntu 18.04LTS (32 bit)
 - CPU: 4
 - Memory: 1GB
 - Disk: 10 GB
 - gcc: 7.4.0
-
- Please follow this [link](#) to install. Normally we will grade your assignments in this environment.
 - Try to Google first when your see any error message.



Other Options to Access Linux

If you really cannot use VirtualBox in your computer:

- CSE Linux server
 - Mac Users
 - i. Open a terminal
 - ii. Type "`ssh YourUnixName@gw.cse.cuhk.edu.hk`", and enter your unix password.
 - iii. Type "`ssh linux2`" to connect to the Linux server.

```
alec@alec-Legion-C530-19ICB:~$ ssh zqwang@gw.cse.cuhk.edu.hk
zqwang@gw.cse.cuhk.edu.hk's password:
Linux nile 3.2.0-4-amd64 #1 SMP Debian 3.2.60-1+deb7u3 x86_64

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Last login: Wed Sep  8 10:28:28 2021 from 137.189.240.39

Department of Computer Science and Engineering
The Chinese University of Hong Kong

| myHome |   L o g i n   g a t e w a y
|-----|
[http://www.cse.cuhk.edu.hk/corner/tech/guide/network/gw]
Enquiry: help@cse.cuhk.edu.hk

* Dedicated to users of the department only.
* No unauthorized access is allowed.
* SSH/rlogin/telnet to internal w/s now!

*** PLEASE USE SSH ***
```

```
<myHome> ssh linux2
zqwang@linux2's password:
Linux linux2.cse.cuhk.edu.hk 3.2.0-5-686-pae #1 SMP Debian 3.2.96-3 i686

The Chinese University of Hong Kong
Department of Computer Science and Engineering

Linux Cluster (32-bit)

-----
Resources limit applied:

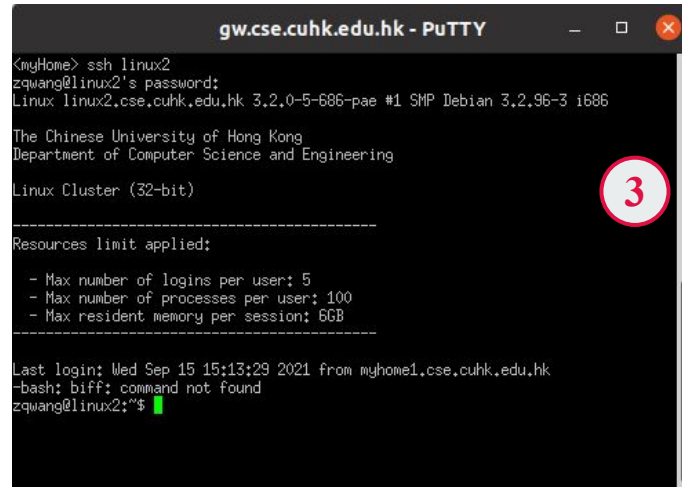
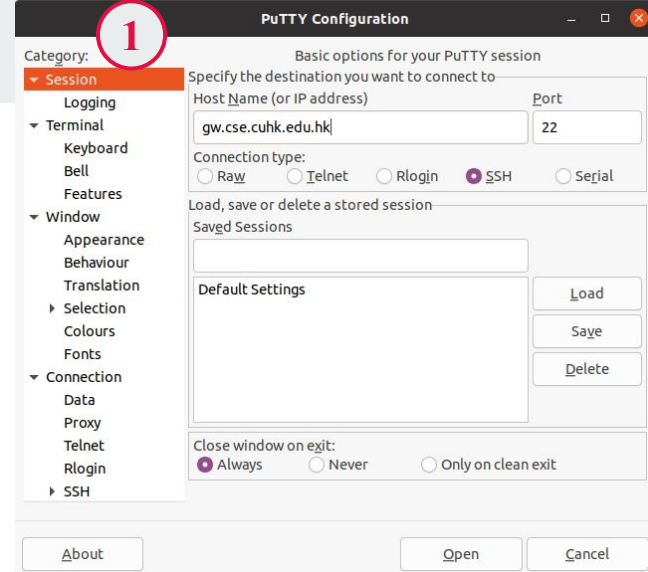
- Max number of logins per user: 5
- Max number of processes per user: 100
- Max resident memory per session: 6GB
-----

Last login: Wed Sep 15 15:12:15 2021 from myhome1.cse.cuhk.edu.hk
```

Other Options to Access Linux

If you really cannot use VirtualBox in your computer:

- CSE Linux server
 - Windows Users
 - i. Install [Putty](#)
 - ii. Follow the steps in the Figures.





Other Options to Access Linux

If you still cannot connect to CSE Linux server, other options may be:

- Install a Linux distribution on your computer (Ubuntu, Debian, CentOS, Arch, etc.)
- Use Docker to access Linux (e.g. There are many materials on the Internet to access Ubuntu with Docker)
- ...

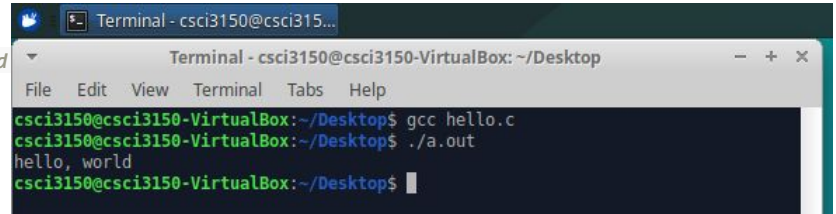
C Programing Review - Compile/Run

```
/* 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
    */
    /* printf returns an integer, but we ignore that */
    printf("hello, world\n");
    /* return 0 to indicate all went well */
    return(0);
}
```

Compile the program:

prompt> gcc hello.c

prompt> ./a.out

A screenshot of a terminal window titled "Terminal - csci3150@csci3150-VirtualBox: ~/Desktop". The terminal shows the following commands and output:

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



Useful flags in gcc

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



Makefile tutorial

```
// hellomake.c

#include "hellomake.h"

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

```
// hellofunc.c

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

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

```
// hellomake.h

void
myPrintHelloMake(void);
```

To compile them:

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




Makefile (first approach)

```
target: dependency1 dependency2 ...
```

```
<tab> command
```

```
hellomake: hellomake.c hellofunc.c
```

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

Suppose we name it Makefile1, then we compile with it:
`make -f Makefile1`

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



Makefile (second approach)

```
CC=gcc
CFLAGS=-I .

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

clean:
    rm hellomake
```

Suppose we name it Makefile2, then we compile with it:
make -f 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!
```



Exercise (Deadline: 2021-09-23 23:59:59)

In the folder exercise, you can find a file **main.c** and two sub-folders: (1) foo; (2) bar. The **main** function in **main.c** will call the functions defined in foo/foo.c and bar/bar.c.

Under the folder exercise, there is a Makefile that will compile the main.c together with foo/foo.c and bar/bar.c and generate an executable file lab1. You need to fill your code in Makefile.

Please only submit the Makefile to blackboard after you finish.