School of Computer Science
Faculty of Science
COMP-2560: System Programming (Fall 2021)

Lab#	Date	Title	Due Date	Grade Release Date
Lab02	Week 02	**Working Environment Setup**	Sept. 29, 2021, Wednesday 4:00 AM EDT	Oct. 04, 2021

The main objective of the first lab will be for you to set up a working environment, specifically to connect to a UNIX-based or UNIX-like operating system (OS) or have it on your own computer.

Step 1. Environment Setup

The labs are the practical sessions of the course “Systems Programming” and comprise 9 manuals that cover hands-on experience with system calls included in the UNIX-based kernel, such as creating multiprocess and multithread programs, inter-process communication, file system and directory access, and sockets and networking. The labs are greatly emphasized to correlate the theoretical concepts of the course with practical applications. We will use C as the programming language.

You can either install a fresh copy of a UNIX-based/like OS such as Linux by Ubuntu[1] or connect to computer systems available in the school to work remotely. Fresh installation of an operating system is not part of this course as all the students can connect to the school’s computer systems with the already Debian GNU/Linux 11. What follows illustrate the connection to such systems depending on:

- Whether you’re connecting from a computer outside or inside the campus (e.g., a laptop that is connected to the school’s wifi vs. from home)
- Your computer’s OS (e.g., Windows, macOS, …)

1.1. Connecting from Outside Campus via VPN

If you’re connecting from home or anywhere outside the campus, you have to connect to the school’s network via a VPN server. You can find help from the school’s user guide at https://help.cs.uwindsor.ca/mediawiki/index.php/VPN or the university’s IT service help at https://www.uwindsor.ca/itservices/talks/installing-globalprotect-vpn. If you’re on campus and connected to the campus’s wifi, there is no need for a VPN connection since you’re already inside the campus computer network.

1.2. Connecting *to* UNIX-based/like Server

A remote connection between a server (host computer system) and a client (guest computer system) can be made through a myriad of client applications[2] that all use a *secure* network protocol called **Secure Shell (SSH)** to provide secure access to the server’s command-line (shell), login, and remote command execution. To download or upload files to the server, however, there are other secure file transfer protocols such as **Secure FTP** or **Secure Copy (SCP).** Client applications may or may not support both protocols. For instance, *old versions* of PuTTY[3] could not do file transfer, and we should install a separate application PuTTY Secure Copy (PSCP)[4] for file transfer. Client applications may or may not have a graphical user interface (GUI). For instance, PuTTY and PSCP do not have GUI, while WinSCP[5] (for Windows) or Cyberduck[6] (for macOS) have GUI. *Fortunately, most current operating systems, either Windows, macOS, or Linux, include SSH and SCP network protocols to create connections. Hence, before installing a new client application, see if you already have them.*

[1] https://ubuntu.com/#download
[2] https://en.wikipedia.org/wiki/Comparison_of_SSH_clients
[3] https://www.chiark.greenend.org.uk/~sgtatham/putty/
[4] https://it.cornell.edu/managed-servers/transfer-files-using-putty
[5] https://winscp.net/eng/download.php
[6] https://cyberduck.io/

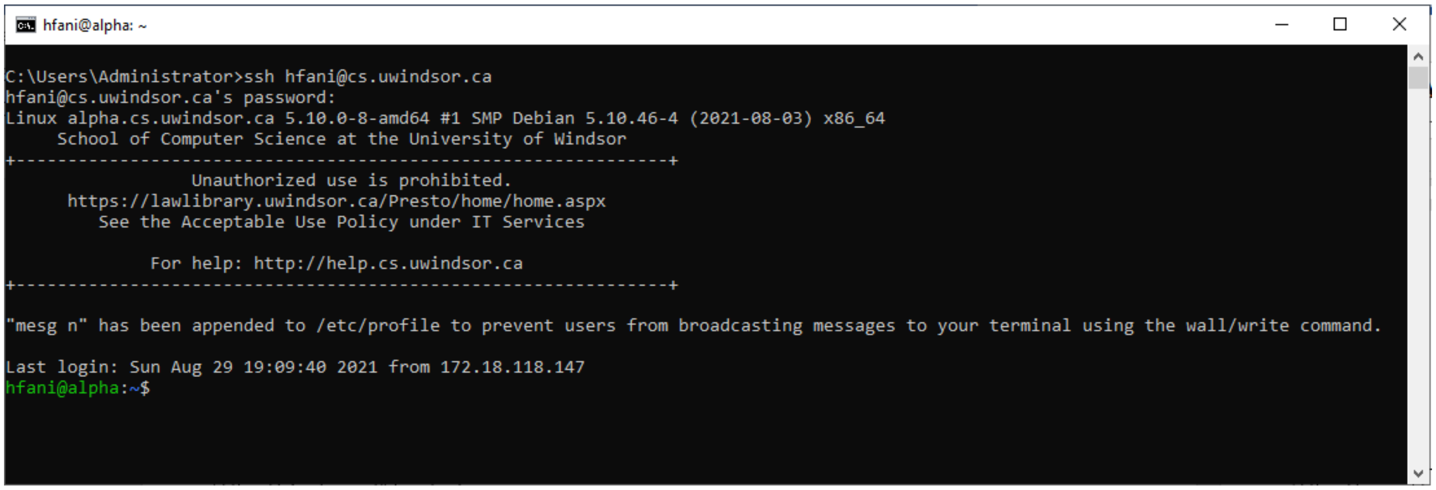
1.3. Connecting from macOS

Mac systems already have SSH embodied in the **Terminal** application.

In **Finder**, open the **Applications** folder, double click on the **Utilities** folder, and double click on the **Terminal** application. Enter the following standard SSH command as follows:

```
ssh uwinid@cs.uwindsor.ca
```

Replace `uwinid` with your own. When asked for a password, put your uwin password. This will connect to the server and the connection will look similar to the following. For instance, my `uwinid` is `hfani`, so I entered `hfani@cs.uwindsor.ca`

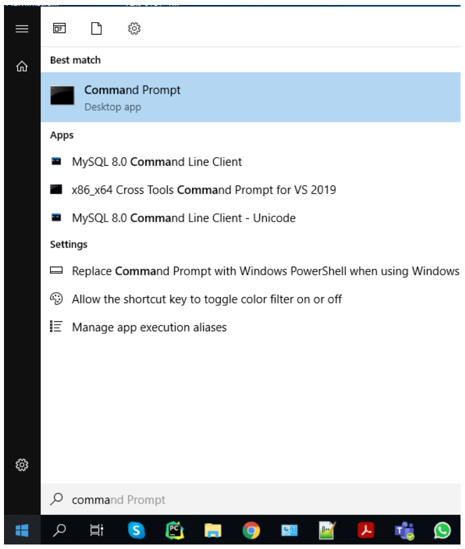


hfani@alpha: ~

```
C:\Users\Administrator>ssh hfani@cs.uwindsor.ca
hfani@cs.uwindsor.ca's password:
Linux alpha.cs.uwindsor.ca 5.10.0-8-amd64 #1 SMP Debian 5.10.46-4 (2021-08-03) x86_64
     School of Computer Science at the University of Windsor
+-----------------------------------------------------------------+
                  Unauthorized use is prohibited.
      https://lawlibrary.uwindsor.ca/Presto/home/home.aspx
         See the Acceptable Use Policy under IT Services

              For help: http://help.cs.uwindsor.ca
+-----------------------------------------------------------------+

"mesg n" has been appended to /etc/profile to prevent users from broadcasting messages to your terminal using the wall/write command.

Last login: Sun Aug 29 19:09:40 2021 from 172.18.118.147
hfani@alpha:~$
```

1.4. Connecting from Windows

Since Windows 10, Microsoft added SSH as a built-in feature. But it is not enabled by default and you have to enable it. In this link, you can find the help: https://www.howtogeek.com/336775/how-to-enable-and-use-windows-10s-built-in-ssh-commands/ . When enabled, open **Command Prompt** by clicking the **Start** button and type in **Command Prompt,** and then select it from the list when it appears:



Best match

Command Prompt
Desktop app

Apps

MySQL 8.0 Command Line Client

x86_x64 Cross Tools Command Prompt for VS 2019

MySQL 8.0 Command Line Client - Unicode

Settings

Replace Command Prompt with Windows PowerShell when using Windows

Allow the shortcut key to toggle color filter on or off

Manage app execution aliases

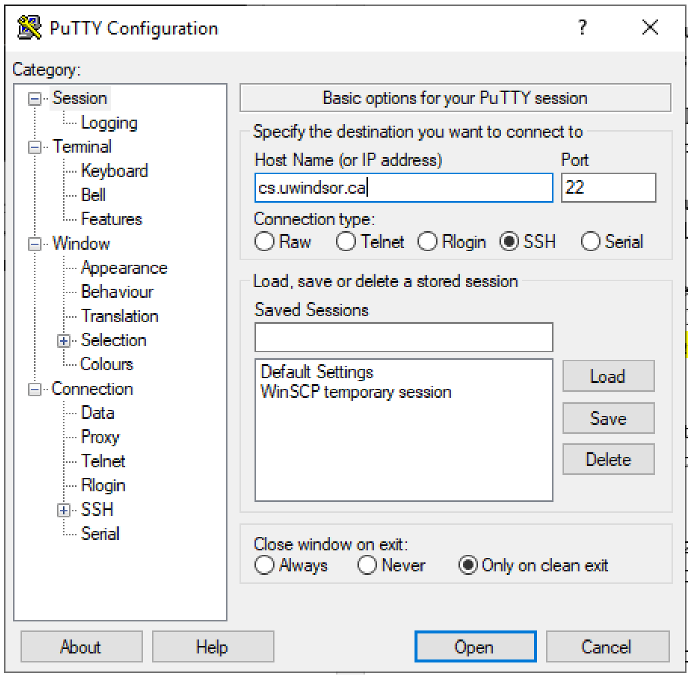
command Prompt

Enter the following standard SSH command as follows:

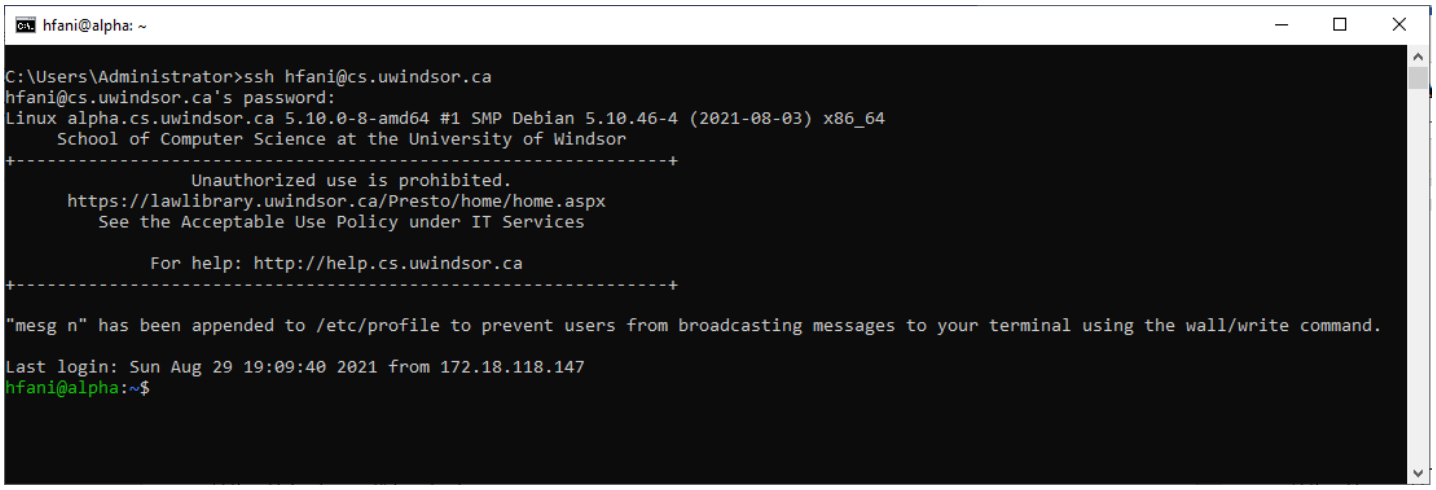
```
ssh uwinid@cs.uwindsor.ca
```

Replace `uwinid` with your own. When asked for a password, put your uwin password. This will connect to the server. For instance, my `uwinid` is `hfani`, so I entered `hfani@cs.uwindsor.ca`

If your Windows misses SSH, you have to install a client application such as PuTTY available at https://www.putty.org/. When you run PuTTY, it looks like the following window to enter the hostname `cs.uwindsor.ca`. The difference here is that PuTTY asks for the username separately. When asked, enter your `uwinid` followed by the password.



Specify the destination you want to connect to
Host Name (or IP address): cs.uwindsor.ca
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions:
Default Settings
WinSCP temporary session
Close window on exit: Always Never [✓ Only on clean exit]



```
C:\Users\Administrator>ssh hfani@cs.uwindsor.ca
hfani@cs.uwindsor.ca's password:
Linux alpha.cs.uwindsor.ca 5.10.0-8-amd64 #1 SMP Debian 5.10.46-4 (2021-08-03) x86_64
     School of Computer Science at the University of Windsor
+----------------------------------------------------------------+
                  Unauthorized use is prohibited.
      https://lawlibrary.uwindsor.ca/Presto/home/home.aspx
         See the Acceptable Use Policy under IT Services

              For help: http://help.cs.uwindsor.ca
+----------------------------------------------------------------+

"mesg n" has been appended to /etc/profile to prevent users from broadcasting messages to your terminal using the wall/write command.

Last login: Sun Aug 29 19:09:40 2021 from 172.18.118.147
hfani@alpha:~$
```

Step 2. Hello World!

Once we can successfully connect to a UNIX-based/like OS, we write our first program in C programing language. To this end, we need the following in order:

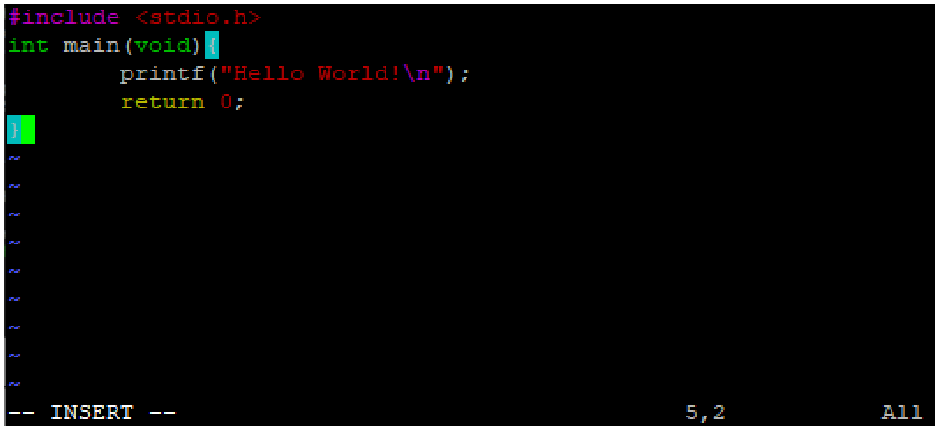
1. A text editor to write our C codes
2. A compiler for C language that translates our C codes to assembly codes
3. An assembler for the host machine to translate the assembly codes to opcodes
4. A linker that includes other required opcodes

2.1. Writing our C Program

vi[7] (/ˌviːˈaɪ/ pronounce v eye) is the most common text editor which is available on almost all UNIX-based/like OS. You can create an empty text file by typing `vi` followed by the filename (`hello.c`):

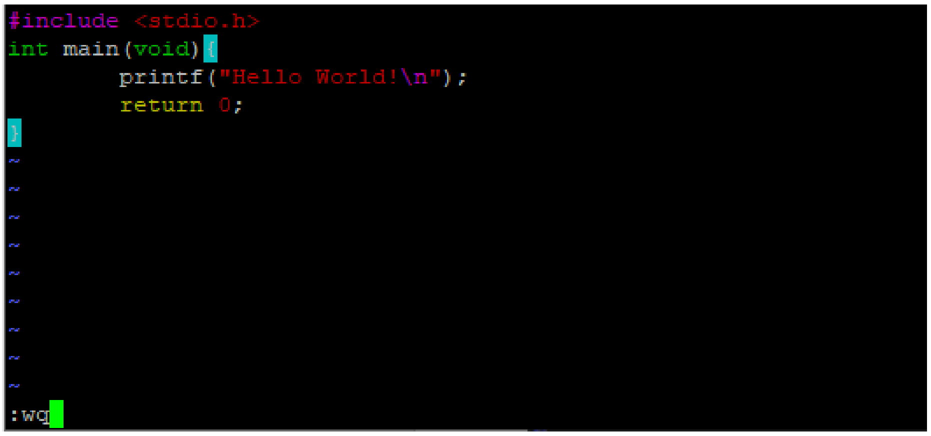
```
hfani@alpha:~$ vi hello.c
```

To start inserting new characters, you should put `vi` in the `--INSERT—` mode by pressing `SHIFT I`. Then start typing the simplest C program as follow:



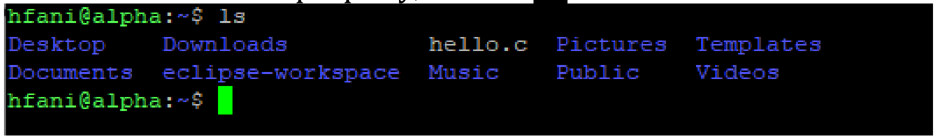
```
#include <stdio.h>
int main(void){
        printf("Hello World!\n");
        return 0;
}
~
~
~
~
~
~
~
~
~
-- INSERT --                                        5,2           All
```

To exit the edit mode, press `ESC` key. In order to save the file, you should press `SHIFT :` after which `vi` needs a command. Enter `wq,` which mean write and quit.



```
#include <stdio.h>
int main(void){
        printf("Hello World!\n");
        return 0;
}
~
~
~
~
~
~
~
~
~
:wq
```

To ensure that your file has been saved properly, use the `ls` command to list the files.



```
hfani@alpha:~$ ls
Desktop     Downloads           hello.c  Pictures  Templates
Documents   eclipse-workspace   Music    Public    Videos
hfani@alpha:~$
```

2.2. Compile our C Program

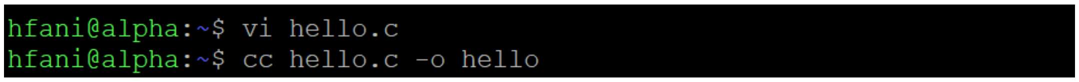
To compile our program, we use the built-in C compiler application, named `cc`, as follows:

7

```
hfani@alpha:~$ cc hello.c -o hello
```

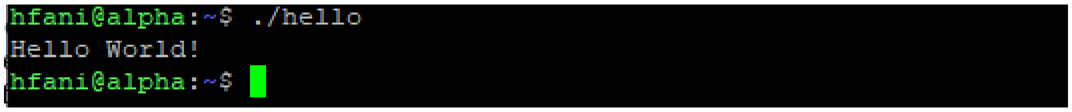
The above command does the following in order:

1. Compile the program file `hello.c` and translate it to assembly language (assembly codes)
2. Translate it to opcodes*!* (binary codes)
3. Link it to other opcodes*!* (e.g., `stdio.h`)
4. Merge all opcodes and create an executable file in `hello` (the option `-o`). If you don't put the `-o` option, it creates `a.out` file as the default filename.



```
hfani@alpha:~$ vi hello.c
hfani@alpha:~$ cc hello.c -o hello
```

And now, the executable file is ready to be launched and run.



```
hfani@alpha:~$ ./hello
Hello World!
hfani@alpha:~$
```

2.3. Create Assembly Codes

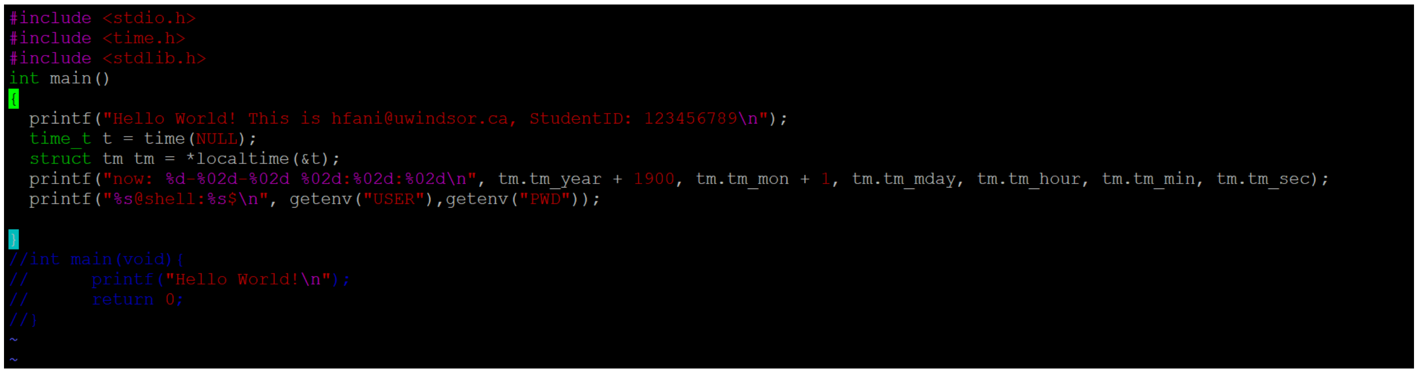
As seen in the previous step, when we compiled our program, it was already translated into assembly codes. In other words, the C compiler does the assembly translation too. If you need more about assembly codes and assemblers, refer to the course COMP2660: Computer Architecture II: Microprocessor Programming.

2.4. Link our C Program

As seen in the previous step, when we compiled our program, it was already linked to the other libraries such as `stdio.h`. In future labs, we introduce advanced topics about linking step.

Step 3. Lab Assignment

You should customize the following C program based on your information (`uwinid` and `student#`) and compile it. Then execute it to make sure it outputs the desired output. The sample code for myself has been attached in a zip file named `lab02_hfani.zip`.



```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int main()
{
  printf("Hello World! This is hfani@uwindsor.ca, StudentID: 123456789\n");
  time_t t = time(NULL);
  struct tm tm = *localtime(&t);
  printf("now: %d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday, tm.tm_hour, tm.tm_min, tm.tm_sec);
  printf("%s@shell:%s$\n", getenv("USER"),getenv("PWD"));

}
//int main(void){
//      printf("Hello World!\n");
//      return 0;
//}
~
~
```

4.1. Download files *from* Server and Upload files *to* Server

As said above, to download files from the server (e.g., your code and executable file) or to upload a file (e.g., the sample code to the Server), you should use an application that implements a file transfer protocol such as SCP, PSCP, WinSCP, Cyberduck, etc. In case you use SCP, the command is as follows:

```
scp source:file target:file
```

As seen, SCP can copy from any source to any target. In case of **download**, the source is the server and the target is our local computer:

```
scp hfani@cs.uwindsor.ca:hello.c C:/Users/Administrator/Desktop/hello.c
```

In case of **upload**, the source is our local computer and the target is the server:

```
scp C:/Users/Administrator/Desktop/hello.c hfani@cs.uwindsor.ca:hello.c
```

Note that the `scp` *should be run from your local computer!*

4.2. Deliverables

You will prepare and submit the program in one single zip file `lab02_uwinid.zip` containing the following items:

(90%) `lab02_uwinid.zip`

- (70%) `hello.c` => must be compiled and built with no error.
- (20%) `results.pdf/jpg/png` => the image snapshot of the output
- (Optional) `readme.txt`

(10%) Files Naming and Formats

Please follow the naming convention as you lose marks otherwise. Instead of `uwinid`, use your own account name, e.g., mine is hfani@uwindsor.ca, so, `lab02_hfani.zip`