

**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** | September 22, 2021 Wednesday 4:00AM EDT | September. 27, 2021 |

The objectives of the first lab will be for you to set up a working environment, specifically to connect to a UNIX-based operating system (OS) or have it on your own desktop or laptop computer. You will be developing on the program specification of the lab assignments that you will be gradually completing throughout this term.

**Step 1. Environment Setup**

The labs are the practical sessions of the course “Systems Programming” and comprises of 9 manuals which cover hands-on experience with system calls included in the UNIX-based kernel such as to create multiprocessed and multithreaded 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 fress copy of a UNIX-based OS such as Linux by Ubuntu[[1]](#footnote-1) or connect to computer systems available in the school to work remotely. Fresh installation of a UNIX-based operating system is not part of this course as we all the students are able to connect to the school’s computer systems with already Debian GNU/Linux 11 operating system installed. What follows illustrate the connection to such systems depending on:

* Whether you’re connecting from a computer outside or inside the campus (e.g., laptop that connected to the school’s wifi or from home)
* The operating system on your computer (e.g., Windows, Mac, UNIX-based OS)
  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 vpn server. You can find help the school of computer science’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 in campus and connected to the campus’s wifi, there is no need for vpn connection since you’re already inside the campus computer network.

* 1. **Connecting *to* UNIX-based Server**

A remote connection between a server (host computer system) and a client (guest computer system) can be done through a myriad of client applications[[2]](#footnote-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. In order to download or upload files to the server, however, there are another 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]](#footnote-3) were not able to do file transfer and we should install a separate application PuTTY Secure Copy (PSCP)[[4]](#footnote-4) for file transfer. Client applications may or may not have graphical user interface (GUI). For instance, PuTTY and PSCP do not have GUI while WinSCP[[5]](#footnote-5) (for Windows) or Cyberduck[[6]](#footnote-6)(for macOS) have GUI.

*Fortunately, most of current operating systems, either Windows, macOS, or Linux, include SSH and SCP network protocols to create connections. Hence, before installing a new client application, make sure if you already have them.*

* 1. **Connecting from Mac**

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

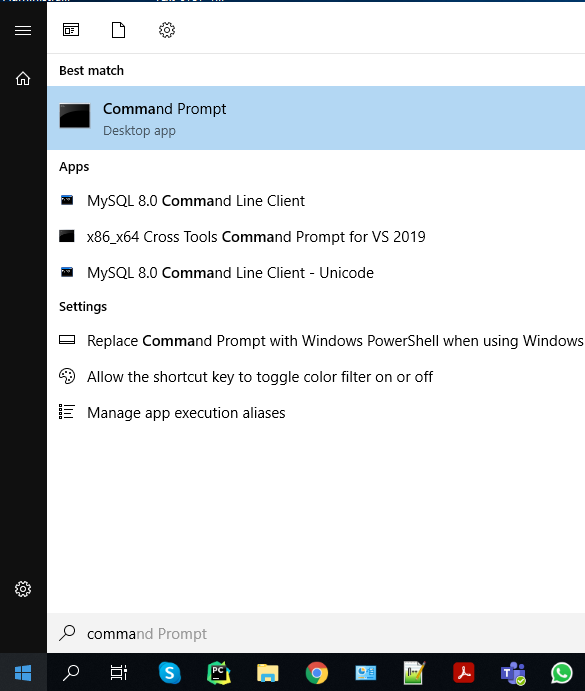
In **Finder**, open the **Applications** folder, double click on the **Utilities** folder, and double click on the **Terminal**application. Enter the following standard SSH command as follows:

ssh *uwinid*@cs.uwindsor.ca

Replace uwinid with your own uwindsor id. When ask for password, put your uwin password. This will connect to the server and the connection will look similar to the following:

* 1. **Connecting from Windows**

Since Windows 10, Microsoft added SSH as 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:



Enter the following standard SSH command as follows:

ssh *uwinid*@cs.uwindsor.ca

Replace uwinid with your own uwindsor id. When ask for 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

Text

Description automatically generated

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 where you have to enter the host name cs.uwindsor.ca. The difference here is that PuTTY ask the username separately. When asked, enter your uwinid followed by password.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

**Step 2. Hello World!**

Once we could successfully connect to a UNIX-based OS, we want to write our first program in C programing language in a UNIX-based OS environment. To this end, we need the followings in order:

1. A text editor to write our C codes
2. A compiler for C language
3. A linker for C language

**2.1. Writing our C Program**

vi[[7]](#footnote-7) (/ˌviːˈaɪ/ pronounce v eye) is the most common text editor which is available on almost all UNIX-based 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-- by pressing SHIFT+I and start styping the most simplest C program as follow:

A screenshot of a computer

Description automatically generated

To exit the edit mode, you should 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.

A screenshot of a computer

Description automatically generated

To make sure that your file has been save properly, use the ls command to list the files.

Text

Description automatically generated

**2.2. Compile our C Program**

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

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

The above command does the followings in order:

1. Compile the program file hello.c
2. Translate it to assembly language (assembly statements)
3. Translate it to opcodes (binary statements)
4. Link it to other opcodes (e.g., stdio.h)
5. 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 and default filename

Text

Description automatically generated

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

Shape

Description automatically generated with medium confidence

**2.3. Link our C Program**

As seen in previous step, when we did compile our program, it’s already linked to the other libraries such as stdio.h. For now, we omit this part and leave it to future labs where we introduced advanced topis.

**Step 3. Lab Assignment**

You should customize the following C program based on you information (uwinid and student id) 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.

Text

Description automatically generated

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

As said above, in order 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 is able to copy from any source to any target. In case of **download**, source is the server and target is our local computer:

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

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

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

* 1. **Delivarables**

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.
* (10%) hello => must be executed with no error.
* (10%) 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](mailto:hfani@uwindsor.ca), so, lab02\_hfani.zip

1. https://ubuntu.com/#download [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Comparison\_of\_SSH\_clients [↑](#footnote-ref-2)
3. https://www.chiark.greenend.org.uk/~sgtatham/putty/ [↑](#footnote-ref-3)
4. https://it.cornell.edu/managed-servers/transfer-files-using-putty [↑](#footnote-ref-4)
5. https://winscp.net/eng/download.php [↑](#footnote-ref-5)
6. https://cyberduck.io/ [↑](#footnote-ref-6)
7. [↑](#footnote-ref-7)