

Lab 1: Connecting to Virtual Servers and Compiling and Testing C/Java Programs on Ubuntu Linux

Part I: Access the Virtual Servers (VS) from a Wireless Network or an Off-Campus Network via MSU Denver VPN

Work on the following tasks on your own laptop that is wirelessly connected on campus or your computer at home:

- A. According to ITS, the **Global Protect** access has already been setup for all students in CS3600. Or you may go to the following link to fill a form to “*Request Access*” to Global Protect. Also, you may also click “*How do I connect to GlobalProtect via the Client*” at the following link to download and install the GlobalProtect Client on your computer.
<https://msudenver.teamdynamix.com/TDClient/2313/Portal/Requests/ServiceDet?ID=46065>
- B. Following the instructions for the operating system on your computer, connect to the MSU Denver GlobalProtect using your MSU Denver NetID and password for student hub, etc. (**Note:** at home, these two steps must be completed BEFORE trying to connect to **cs3600a** or **cs3600b**.)

Warning: ITS only supports GlobalProtect on **MAC** and **Windows** machines. If your home computer has a different OS, it is your responsibility to figure out how to connect to cs3600a and/or cs3600b for programming assignments and submit your work by the cutoff deadline. Not being able to connect to GlobalProtect or virtual servers (cs3600a and cs3600b) can NOT be used as an excuse to request any extension of homework due date or cutoff deadline.

Part II. Access the Virtual Servers on-campus or after connecting to GlobalProtect (A: MAC/Linux, B: Windows)

- A. **If your computer is a MAC or Linux computer**, you can open a **Terminal** window to work on the following tasks.
 1. Use the following command to connect to **cs3600a.msudenver.edu** and **cs3600b.msudenver.edu**, for **remote operations**
`ssh <your MSU Denver NetID>@cs3600a.msudenver.edu (or cs3600b...)`
 - Once connected, you will be asked to input the password. The initial password is ABCDWXYZ, where ABCD is the last four digits of your 900 number and WXYZ is your birthday in the MMDD format.
 - After you type the initial password, you will be asked to change the password. Go ahead change it and make sure that you will memorize it.
 - After you change the password, you might be disconnected. Just go ahead use **ssh** for re-connection.
 - While connecting to a virtual server, the following commands could be useful

```
mkdir <name of the directory> //to make a new directory
cd <name of the directory> //change the directory
pwd //display the current directory
ls -l //list the files in the current directory
cp <name of the file> <name of the directory> //copy a file to a directory
mv <name of the file> <new name of the file> //rename a file
rm <name of the file> //delete a file
rm -r <name of the directory> //delete a directory
vi <name of the file> //use the text editor vi to edit a text file
man <name of the command> //display the manual of the command
```
 2. Use the following command to connect to a virtual server, cs3600a.msudenver.edu (or cs3600...), for **file transfer** (**Warning: a new terminal on your local MAC or Linux computer needs to be open for this command!!!**)
`sftp <your MSU Denver NetID>@cs3600a.msudenver.edu (or cs3600b...)`
 - Once connected, you will be asked to input the password. Use the password you set in the above step.
 - After you type the password, you will see a prompt, **sftp>**. Now, there is an **sftp** channel connecting the MAC/Linux computer, the *local* end, and the virtual server, the *remote* end. The following commands could be used for data transfer in the command-line window with the **sftp>** prompt.

```
!pwd //display the current directory on the local end
!ls -l //list the files in the current directory on the local end
lcd <name of the directory> //change the directory on the local end
pwd //display the current directory on the remote end
ls -l //list the files in the current directory on the remote end
cd <name of the directory> //change the directory on the remote end
put <file name> //upload a file from the local end to the remote end
```

```

get <file name>    //download a file from the remote end to the local end
mput *.c          //upload all .java files from the local end to the remote end
mget *.c          //download all .java files from the remote end to the local end
quit              //close the sftp channel (disconnect)

```

B. If your computer is a Windows computer, you can download **PuTTY**, a free SSH client for Windows, and **PSFTP**, a free command-line SFTP tool for Windows. Then configure and use **PuTTY** and **PSFTP** for the following tasks using their online documents for reference.

1. Once connecting to a VS via **PuTTY**, all commands in **Part II.A.1.** can be executed for you to work on a VS.
2. Once the **PSFTP** window is open on **your local Windows computer**, run the following “**open**” command to establish a sftp connection with a virtual server. Then all the commands except for “**!ls**” and “**!pwd**” in Part I Step 2 can be executed for you to transfer files between your local computer and a VS.

```

open <your MSU Denver NetID>@<cs3600a or cs3600b>.msudenver.edu //connect
!dir -l                  //list the files in the current directory on the local end

```

Part III. Compile and Test C programs on cs3600a

- Connect to **cs3600a.msudenver.edu** using the information given in Parts I & II. In total, make one **sftp** or **PSFTP** connection and one **ssh** or **PUTTY** connection.
- Download file “*newproc-posix_Zhu.c*” in “a C program for forking a child process” under the “Resources and Examples for C programming” module in Canvas.
- In the **ssh** or **PUTTY** connection, create a directory, “**Lab01**”, on **cs3600a.msudenver.edu** in your home directory.
- In the **sftp** or **PSFTP** connection, upload *newproc-posix_Zhu.c* to “**Lab01**” on **cs3600a.msudenver.edu**.
- In the **ssh** or **PUTTY** connection, use the following commands together with the ones given in Step A.1 or B.1 of Part II for compiling and testing a C program running on a virtual server.

```

gcc <name of the .c file> -o <name of the output file> //compile a .c file
//e.g., gcc newproc-posix.c -o newproc-posix_Zhu
./<name of the output file> //run an executable file that's in current
// directory, e.g., ./newproc-posix_Zhu
ps -ef |grep <name of the program> //list info for process(es)
kill -9 <Process ID of a process> //kill a process with given PID

```

Part IV. Compile and Test more C programs on cs3600a

- You may use a **vi** command on **cs3600a** to edit more .c files and compile them for testing. The code for some of the problems in HW2/HW3 is a complete .c program. You are highly recommended to add *printf()* and/or *fgets()* statements properly into the code, compile it, and run it to verify your solutions.
- You may also study how to define and create an empty linked list first, create a new item and insert that item to a linked list, remove and delete/destroy an item from a linked list, and update the header and tail pointers accordingly in your C program. You may refer to the following link (but the “Execute Code” is NOT good) or any other resource on C programming. It’s a good idea to **typedef** the **struct** (an **item** to be linked into the list) outside **main()** or any function.

http://www.learn-c.org/en/Linked_lists

Part V. Compile and Test Java programs on cs3600a

- The following commands are for compiling and testing Java programs on **cs3600a** or **cs3600b**

```

javac <name of the .java file> //compile a .java source code file
//e.g., javac HelloWorld.java
java <name of the .class file w/o extension> //run a java program
//e.g., java HelloWorld

```