

Connecting to UNIX/LINUX at NJIT

Alex. Gerbessiotis
CS Department
NJIT

May 25, 2023

A previous document dated 2021 or 2022 such as the ones listed below are superseded by this new document. Many typos have been corrected. It is also consistent with Windows 11 in addition to Windows 10.

[Ref1] **Connecting to *UNIX at NJIT, December 23, 2022.**

[Ref2] **Connecting to *NIX at NJIT, December 23, 2021.**

This 2023 document is Part A of a longer document consisting of four parts of a Brief on Unix/Linux at NJIT.

PART A

Connecting to *NIX at NJIT

Precondition: You have an NJIT-sanctioned laptop or desktop computer that satisfies NJIT computer policies as applicable to the CS Department and/or other NJIT units. As of this writing, you may find the policy at the link below (URL: Uniform Resource Locator) and can be summarized as follows: you have a Windows-based computer (Windows 10 or may be a later version such as Windows 11).

<https://ist.njit.edu/student-computers>

1. OSL (Open System Lab)

The term OSL at NJIT denotes Linux machines physically located on the 2nd floor of the GITC building in a laboratory that is known as Open Systems Laboratory (OSL in short).

The machines located in OSL have DOMAIN names oslX.njit.edu or oslXY.njit.edu, where X,Y are digits mapping to integers from 1 to around 31.

A given domain name maps to a unique IP address starting with 128.234.44.51 for osl1.njit.edu, but note that osl31.njit.edu might map to 128.234.44.47, which is non-intuitive.

In this document, for the sake of the discussion to follow I will be using example machines osl7.njit.edu and osl21.njit.edu. Therefore, X=7 in the former case and X=2 and Y=1 in the latter case.

If you plan to connect to any one of those machines by visiting NJIT's GITC building, skip Section 2 below. If you are inside NJIT, and you are using a wired connection skip Section 2 below. Some URLs (Uniform Resource Locator) of interest are as follows.

0. NJIT computer policies as applicable to the CS Department and other NJIT units are available, as of this writing, at <https://ist.njit.edu/student-computers>
1. The URL for downloading NJIT's copy of MobaXterm, a Windows secure shell client (ssh) is shown below. The URL provides also information about using ssh on Mac OSX and Linux. We do not discuss these options. One may also download a copy of MobaXterm of limited functionality directly from the manufacturer. In that case you may not even need to install MobaXterm; for more see manufacture/publisher's web site. <https://ist.njit.edu/accessing-afs>
2. The URL for 'accessing AFS' which means connecting to a linux machine at NJIT is shown below <https://ist.njit.edu/afs>
3. The NJIT VPN URL with links to downloadable VPN clients for Windows, MacOSX and Linux with instructions is shown below. <https://ist.njit.edu/vpn>
4. An NJIT URL with info on *nix commands is shown below. <https://ist.njit.edu/common-UNIX-commands>

2. VPN

The discussion below uses a client computer that is a WINDOWS 10 machine. This is in accordance with YWCC guidelines. (It also applies to Windows 11 machines.)

NJIT persons have three options in connecting to an OSL NJIT machine:

- (a) Visiting the OSL Lab on the 2nd floor of GITC,
- (b) Using within NJIT a laptop with a wired internet connection,
- (c) Using within NJIT a laptop with a wireless internet connection,
- (d) Using from OUTSIDE of NJIT a laptop.

The discussion to follow applies only to case (d) and in some rare cases to case (c) and both would be referred to as being 'outside of NJIT'. You may skip this section if your situation is under case (a) or case (b).

If you plan to connect to an OSL machine from **outside of NJIT** you must

- (1) Detect if a VPN client has been installed previously on your machine. We expect Computing students can figure out whether a VPN client is preinstalled or previously installed (by you) on their machine (Setting->Programs or Setting->Apps might provide some information). If a VPN client is not installed, first download such a VPN (Virtual Private Network) client though URL 3 of Section 1, and install it. Installation is done once and might require a reboot or a restart.
- (2) Activate VPN if a VPN client is preinstalled on your machine but is currently deactivated (this is a rare case, since by default it is activated at boot time). If you know how to deactivate it, you should know how to activate it or reactivate it.

Instructions below are for a Windows 10 machine but also applicable to a Windows 11 machine.

On my laptop the windows taskbar is at the bottom of the screen, with a Windows icon on the bottom left corner and the time and date information on the bottom right area/corner. In

the bottom right area of the taskbar you might see the icon shown below. If not, find an up-arrow in the right area of the task bar, click on it and see if the icon is depicted on the popped up window. For me, it appears as shown below in Figure 1. This means that the VPN client is **ACTIVATED but it is not in USE ('not connected to the NJIT' network)**. If the VPN client were in USE, the icon would have appeared as in Figure 4 instead.

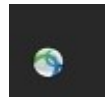


Figure 1.

1. You may click on this icon (shown in Figure 1) and the following pop-up window might appear (Figure 2).

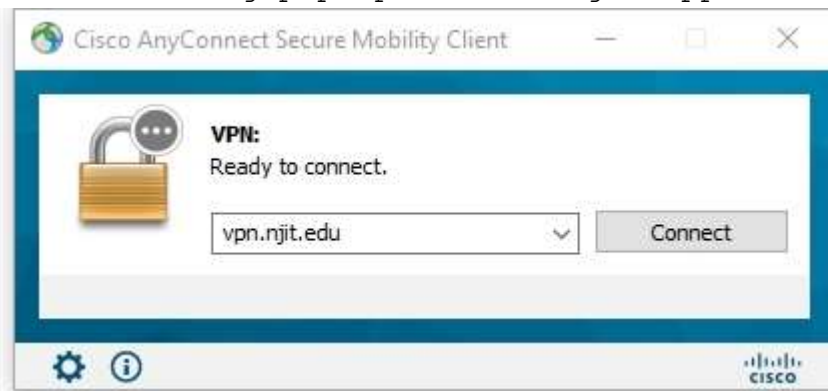


Figure 2.

2. The Connect message (Figure 2) on the button indicates not only that your machine is disconnected (i.e. the VPN client is not in use), but also indicates that your clicking on it will allow a connection to take place and thus you would be able to start using VPN. You are about ready to click Connect. In the next step you need to have ready your myUCID credentials (login and password) which must not have expired. After you click connect the pop up window of Figure 3 is shown.

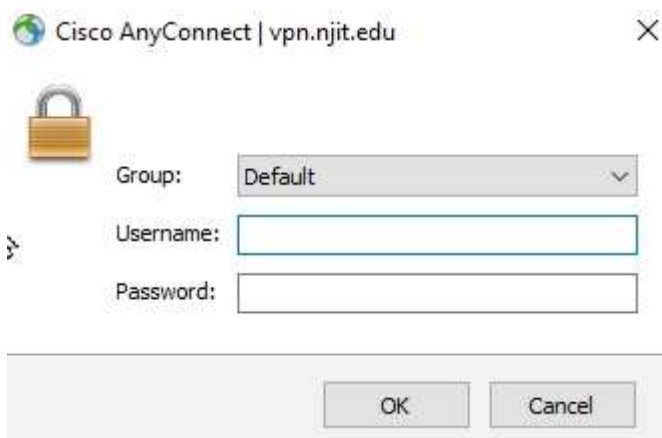


Figure 3.

3. The Username: field might have been pre-populated with your UCID. If not, type in your myUCID login in the Username: field, and then type in your myUCID password in the Password: field. The use of the TAB button (or SHIFT-TAB button) of the keyboard can help you moving around quickly. The Group: option can be left Default or you select an alternative according to the user instructions available during the VPN installation process or other information provided by NJIT.
4. If you have supplied the correct information (correct login name and correct associated password), a connection will be established and VPN would be in use and the popup window of Figure 3 will disappear. At that point if you try to locate the VPN icon using the instructions prior to step 1, the icon has a lock on it as shown below in Figure 4.



Figure 4.

5. If there was an active connection prior to step 2, and there will be an active connection after step 3, the window of Figure 2 would have looked like Figure 5. The button reads Disconnect since VPN is in use vs the Connect in Figure 2 when VPN was not yet in use. You can click on Disconnect to terminate the VPN connection when it is of no need any more. An alternative is to locate the icon shown in Figure 4, click on it, and it will allow you to disconnect and thus terminate the IN-USE VPN session.



Figure 5.

At this point if you are outside of NJIT you have a VPN session in use. The next section deals with the interaction with a secure shell (ssh) client and how it can be used to connect to an OSL machine.

3. Secure Shell Client

By now, you are either in Case 2(b) or 2(c) or 2(d). If you are in case 2(a) you do not need a ssh client, since you can login directly to an OSL machine and each OSL machine has its own one preinstalled. If you are in Case 2(b) you have skipped the rest of the discussion of Section 2. In case 2(c) which is rare or case 2(d) you have completed successfully step 3, and you have a VPN session that is in-use (and of course active).

This means you have an authenticated, network connection-based presence at NJIT.

Download and Install a Secure Shell client (ssh) utilizing URL 1 of Section 1. An alternative is through URL 2. OSX and Linux machines have one pre-installed. The one available for Windows has file transfer capabilities using a graphical interface. Some NJIT provided machines also have it preinstalled. Thus for a Windows machine you may install the secure shell client known as MobaXterm, available to all at NJIT by NJIT, or go to the commercial MobaXterm web-site and download the limited feature free version found there. NJIT provides some info on MobaXterm through the link below that also includes info for secure shell to OSX users.

<https://ist.njit.edu/how-connect-afs-mobaxterm>

The discussion below is for a Windows client using MobaXterm.

1. Invoke MobaXterm. An icon on your desktop might be available for clicking on it. A window as shown in Figure 6 will pop up. Depending on your settings and customizations, it might look different from the one in Figure 6. Click on the button with the message **Start Local Terminal**

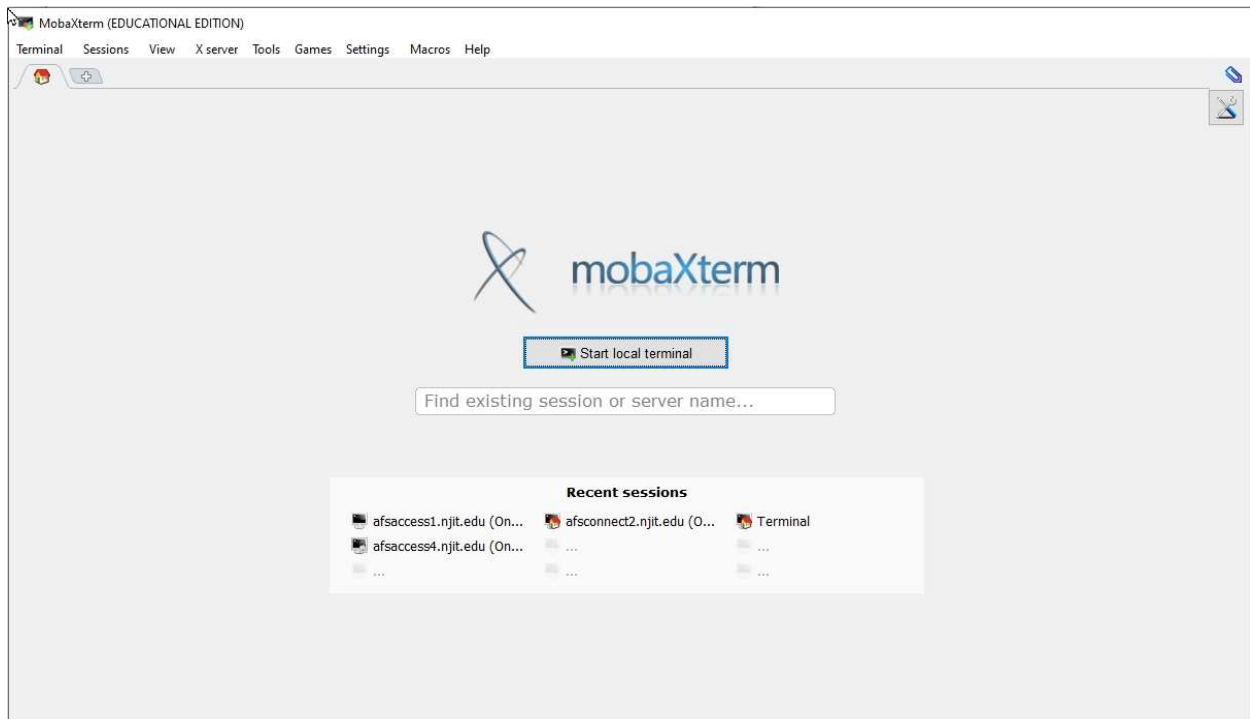


Figure 6.

2. After clicking the button "Start local terminal", a window like the one shown below in Figure 7 will pop up. That window has a **varying prompt (text string)** that ends with a right arrow and then next to the right arrow you may see a blinking cursor in the form of a rectangular box. This is a window running on your local computer, a client application. The prompt is a request by the application for you to provide input. We might call the application a shell. The shell is running on the client machine (your Windows laptop for example).

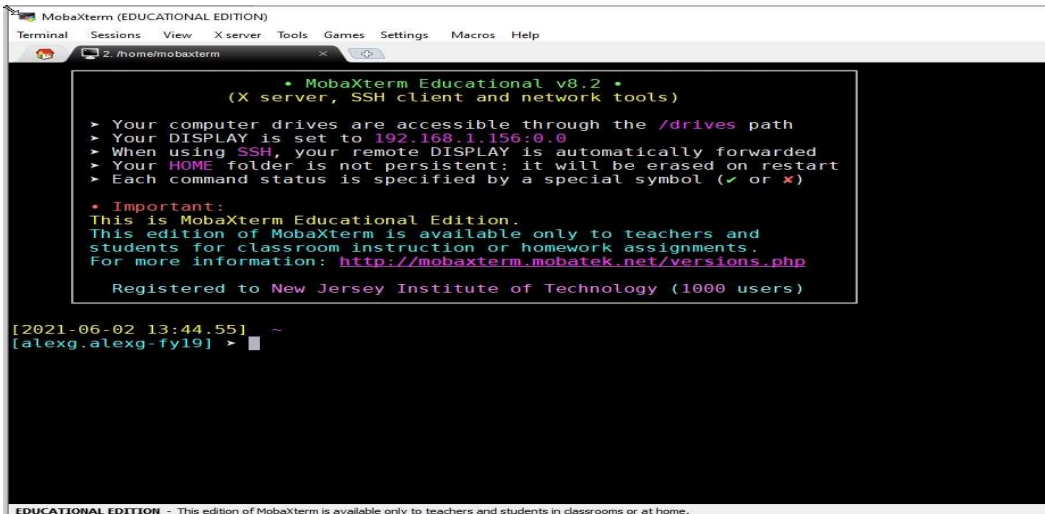


Figure 7.

3. Start writing using the keyboard a shell command that will be read by the client application (shell) of your Windows client machine. The command that you will type will be read by the client application (shell) and executed. The client application will then establish a Secure Shell connection to a remote OSL machine as specified by you in the typed command. As soon as a connection is established you would be writing down commands apparently on the client, but the window would be hosted by the remote host (machine/server), and the commands would be read and executed by the remote host instead.

For the example to follow we pick as a remote host (server) `osl7.njit.edu`.

The syntax of what you should type starting at the cursor's position would be

```
➤ ssh myUCID@host-name
where
```

`ssh` is the name the client program (shell); `ssh` stands for secure shell.

`myUCID` denotes your UCID login name and replace the string accordingly, and

`host-name` is the remote host name to which you want to connect. For this example `host-name` is `osl7.njit.edu`

Do not forget to press ENTER at the end of the typed line and every typed line.

A Warning message might be generated the first time you connect to this host and a password prompt is output for you to provide your myUCID password. The blinking cursor box is waiting for your myUCID password. Type it in and

Do not forget to press ENTER at the end of the typed password.

See Figure 8 below.

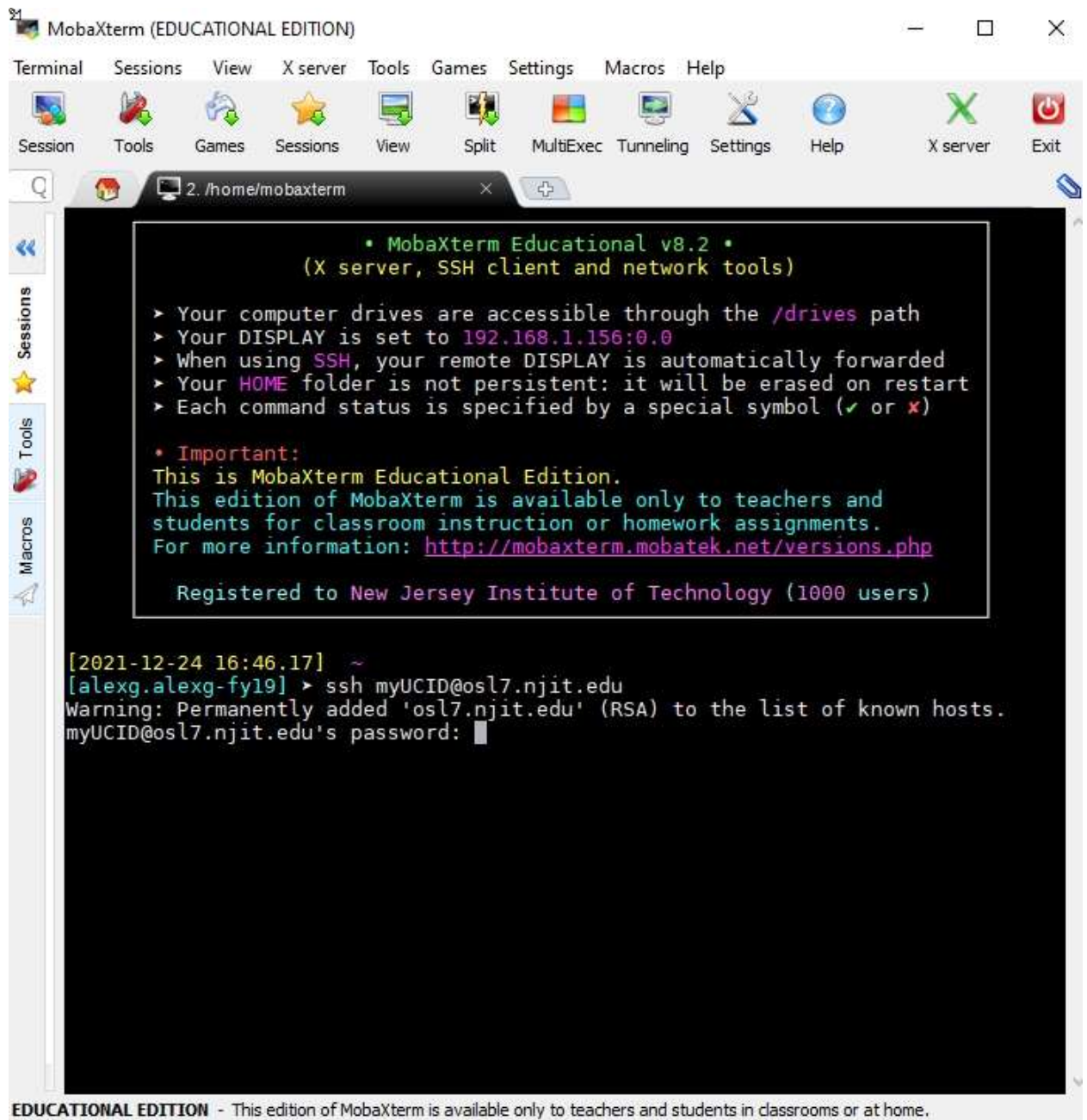
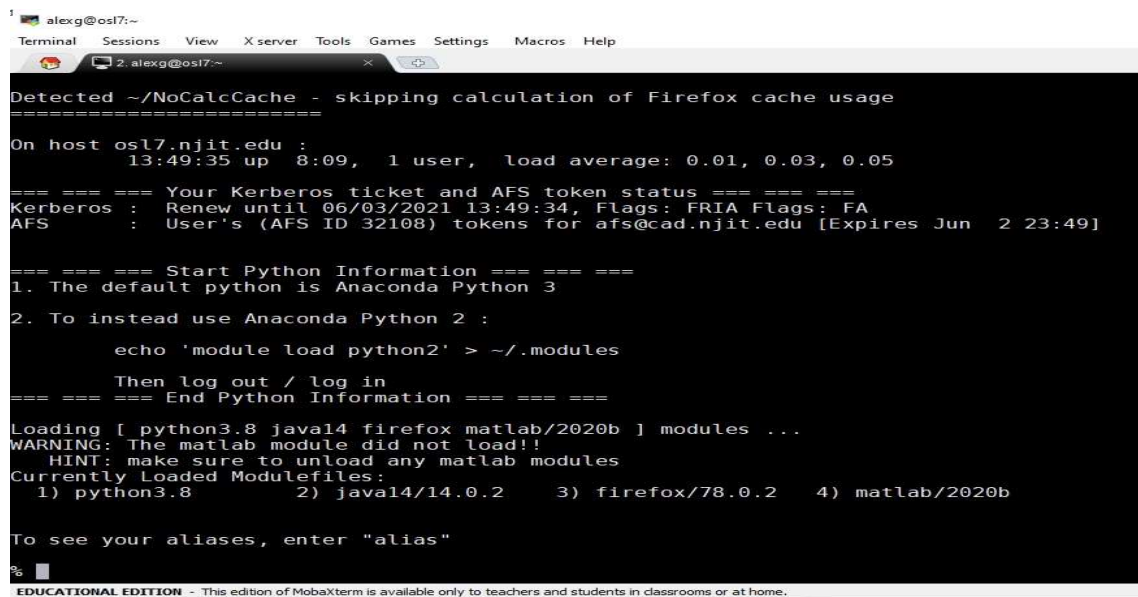


Figure 8.

4. After you typed your password and pressed ENTER you are logged on to the machine in question and of the given host-name). The screen might or might not look like the one in Figure 9. Looking at Figure 9 in the third line from the top you might read a "On host osl7...." verifying that you indeed got connected to osl7.njit.edu as intended.

The window of Figure 9. is a window hosted by the remote host (osl7.njit.edu). Although you will be typing commands in the client machine, your input will be transmitted to, read and executed and interacted by the remote host!

The remote host window has a different prompt as well. At the very bottom of the screen that is Figure 9 you see the prompt which FOR MY OWN SETUP (but maybe NOT FOR YOU) is the percent symbol % and after a space you may see the blinking cursor waiting for your input . The prompt and the cursor are customizable.



```
alexg@osl7:~  
Terminal Sessions View X server Tools Games Settings Macros Help  
2. alexg@osl7:~  
Detected ~/NoCalcCache - skipping calculation of Firefox cache usage  
=====  
On host osl7.njit.edu :  
13:49:35 up 8:09, 1 user, load average: 0.01, 0.03, 0.05  
==== === Your Kerberos ticket and AFS token status ==== ===  
Kerberos : Renew until 06/03/2021 13:49:34, Flags: FRIA Flags: FA  
AFS : User's (AFS ID 32108) tokens for afs@cad.njit.edu [Expires Jun 2 23:49]  
==== === Start Python Information ==== ===  
1. The default python is Anaconda Python 3  
2. To instead use Anaconda Python 2 :  
    echo 'module load python2' > ~/.modules  
    Then log out / log in  
==== === End Python Information ==== ===  
Loading [ python3.8 java14 firefox matlab/2020b ] modules ...  
WARNING: The matlab module did not load!!  
HINT: make sure to unload any matlab modules  
Currently Loaded Modulefiles:  
 1) python3.8      2) java14/14.0.2   3) firefox/78.0.2  4) matlab/2020b  
To see your aliases, enter "alias"  
%   
EDUCATIONAL EDITION - This edition of MobaXterm is available only to teachers and students in classrooms or at home.
```

Figure 9.

You might explore other options of MobaXterm. Moreover it is possible to upload files (from the client machine to the remote host) or download files (from the remote host to the client machine). You might see on the left side or the right side of the MobaXterm window, the file system directory area for your account. Read the Mobaxterm manual

or instructions as made available by NJIT or through the MobaXterm web-site for more information.

At this point you are ready to start interacting with the remote host.

Type in Unix/Linux commands, when done typing them, press the ENTER button and observe the output.

If you are not familiar with Unix or Linux there are several tutorials or summaries out there. Pick the one you are more comfortable with it.

Finally, to terminate the session type exit. You will 'logout' from the remote host, and you will the move back to the familiar screen of Figure 7.

If you are done with MobaXterm, terminate it by clicking on its top right corner the X symbol that will 'kill' the window. Likewise locate the icon of Figure 4 and stop VPN. The icon of Figure 4 will revert to its form in Figure 1. At that point if you rightclick on Figure 1 you will be able to deactivate VPN. Deactivating VPN is NOT EQUIVALENT to an uninstall.

4. Limited Unix command overview

In a Unix system, a user is logged on to the system by providing a set of credentials (login name and an associated password), a process already familiar to you from the earlier sections of this document. At NJIT we call the credentials MyUCID credentials consisting of a myUCID (login name) and a myUCID password.

Immediately after the login has been completed successfully the Unix environment would become available to the logged on user and a program (process) would start executing in the user's environment after the user's login: the Unix shell process.

The UNIX shell process would allow the user to interact with the operating system and start, stop, suspend and resume the execution of services provided by the OS or create and manage user created processes. These services are programs and when run they become processes. The definition of a process is 'a program in execution'. All interaction is done through the terminal and its associated keyboard that was used by the user to gain access to the system: the user types in commands to the shell and after the shell interprets these commands, it invokes services of the OS to execute/realize those user commands as needed and as privileged. The OS might decline to execute some of these commands for safety or security reasons based on the credentials (privileges) of the user.

To keep interaction short UNIX commands to the shell are short and sometimes intuitive. For example command **ls** is short for list, **cat** for catenate (list contents), **mkdir** for make directory, **ps** for process status, **cd** for change directory, **pwd** for print working directory, **mv** for move, and **cp** for copy.

Moreover, the UNIX shell provides command line editing, and history of interactions with the shell thus allowing for editing a previously lengthy command instead of retyping it before re-execution or repeating a frequently executed command by easily recalling it from a history of prior interactions.

Every command of the shell such as **ls**, **ps**, **mv**, **cp**, **pwd**, **cat**, **cd**, is an executable program residing in secondary memory (disk). It was originally written in C and compiled and assembled

subsequently into the executable file named `ls`, `ps`, etc. Thus typing a command such as

```
% ls
```

would cause the shell (an OS process) to load the program named `ls` from secondary memory into main memory, thus turning it into a process and then executing it as needed.

A reminder: the `%` is the shell prompt. You do not type it. It is output by the shell to remind you that 'I, the shell, have your full attention: please type in your request'. Moreover when you do so and type your request (`ls` in this case) do not forget to tell the shell that you are done when you are done typing your command. You do so by pressing the ENTER key of the keyboard at the end.

At that moment the shell interprets your input (in the example above it is an `ls`), the text between the prompt and the ENTER, and executes it as needed. Every execution of a process in linux (in this case `ls`) by default creates and interacts with three files associated and connected with three devices:

- (a) standard input also known to the user as the file with fd (file descriptor) 0,
- (b) standard output with fd equal to 1, and
- (c) standard error with fd equal to 2.

Unless the shell is instructed otherwise by you, standard input is associated with your terminal's keyboard, and standard output and standard error are both associated with the terminal's screen.

Moreover in Linux, multiple commands can be executed one after the other in the command line. Thus

```
% ls ; date ; echo "Hi"
```

Thus in the above example after `ls` is executed, the current date and time is printed, and afterwards a message gets printed on the standard output, the terminal used by the user.

Moreover in Linux, multiple commands can interact with each other with a mechanism known as an unnamed pipe. A pipe is a FIFO (First In First Out) queue that accepts input from the output of one command and generates output that will become the input of another command. Thus


```
% ls | egrep filename
```

consists of the command `ls` that prints the contents of the current directory (this description makes sense after you read the next section if you are not familiar with any operating system's structure) and directs this output to the unnamed pipe indicated by the pipe `|` symbol. The unnamed pipe indicated receives as input the transmitted by `ls` output and then it generates its own output that is to become the input of the command `egrep`. The command `egrep` filters its input by discarding all lines that do not contain the string/word `filename` and thus preserving to the output that it will generate the lines that contain the string `filename`. The combined execution thus prints the output lines of `ls` that contain the string `filename`.

Pipes can allow multiple cascade communication such as the following one.

```
% ls -l | egrep filename | sort | less
```

A list of useful commands

```
% pwd          #print current working directory
% cd path      # change the current working directory to path
% cd          # if you are lost go to (i.e. cd) to home
directory
% cd /         # goto the root of the filesystem
% man pwd     # manual page for command pwd
% man cd      # manual page for cd
% whoami
% who
% date
% hostname
% cal         # calendar; use also eg. cal 5 2023
% stat filename
% ls path     # list contents of directory path
% ls filename # confirm whether filename exists
% ls         # list contents of current working directory
```

```

% ls .          # same as ls; . alias for current directory
% ls ..        # list contents of parent directory
% ls -l        # equivalent to ls -l .
% ls -l filename
% ls -l directory      %ls -l path
% ls -a          # Hidden files . and .. included
% ls -lai       # the inode (numeric name of the file) is also
listed on the extremment left.
% ls -lR        # R for recursive ; r means reverse!
% ls file*      # list all files starting with file
% ls *file      # list all files ending with file
The characters after a dash are option of the command (eg. ls)

% rm file      # delete a file

or

% rm pathToFile

but

% rmdir directory      # remove an empty directory
% rmdir pathToDirectory # remove an empty directory
% mkdir newdirectory   # create a new directory
% rm -rf file          # A pretty dangerous command... Avoid it.
% rm -rf directory    # A pretty dangerous command... Avoid it.
% mkdir directory     # Create a directory
% mv oldname newname  # rename file
% mv olddir newdir    # rename a directory (newdir must not
preexist)
% cp file1 file2      # make a copy of file file1

% cat filename        #list contents of filename

```

```
% more filename          #similar to less
% less filename          # controlled flow version of cat
# head filename
# head -10 filename      # print 10 first lines of filename
% tail -10 filename      # last 10 lines
% egrep me file          # search for me in file
% echo string            # string on screen
% wc filename            # num of chars, lines, words in file
% ps                     # process status
% ps -ef |egrep myID
% sleep 10               # screen freezes for 10sec
```