ESE 566A Modern System-on-Chip Design, Spring 2017
Tutorial 1: Linux Environment

Electrical and System Engineering, Washington University

Revision: 01-05-2017

1. Introduction

All the lab assignments for this course are designed assuming you will be using a Linux (or UNIX like) operating system for development. Basic Linux knowledge is essential to successfully complete these lab assignments and a more in-depth understanding enhances productivity. This tutorial covers the computing resources to be used in the course and offers a brisk introduction to the Linux operating system for first time users including some details specific to this course. To follow along with the tutorial, type the commands without the % character.

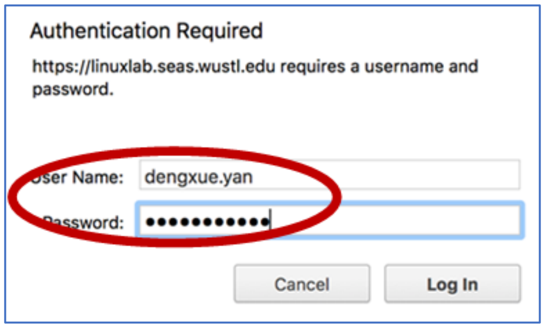
2. Linux Lab server

We will be using Linux Lab servers for all of the laboratory assignments. The Linux Lab machines all run the CentOS 7 operating system. You will be using your Webstac ID and password to login. Any student enrolled in any ESE class should automatically be granted access to the Linux Lab servers. We provide two methods to login to the Linux Lab server.

2.1. Login to Linux Lab server using equeue

This is a graphical login method. Enter the following website in your browser(For example: Chrome) :

- https://linuxlab.seas.wustl.edu/equeue/



Authentication Required

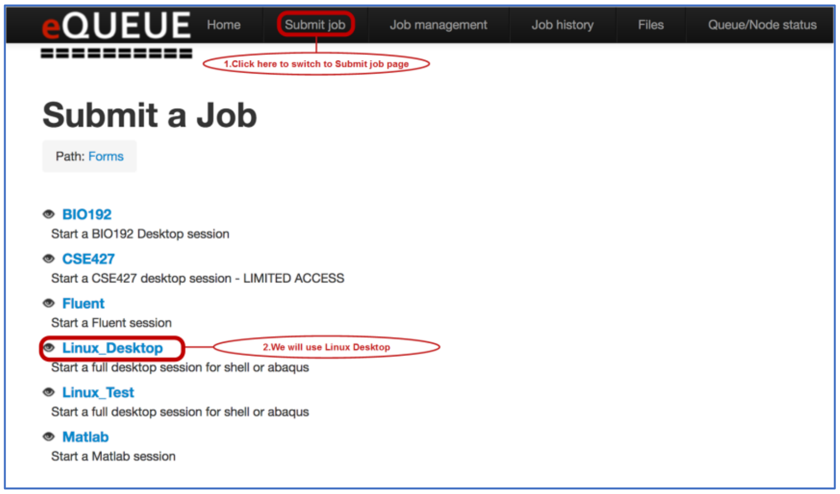
https://linuxlab.seas.wustl.edu requires a username and password.

User Name: dengxue.yan

Password: ••••••••••••

Figure 2.1 Equeue login

When you are asked for User Name and Password, put your Webstac ID and Webstac password(Fig. 2.1). Then click “Log In”, you will get the page like Figure 2.2. First, click *Submit job* button at the top of the page, then select “Linux_Desktop”


eQUEUE
Home
Submit job
Job management
Job history
Files
Queue/Node status
1.Click here to switch to Submit job page
2.We will use Linux Desktop

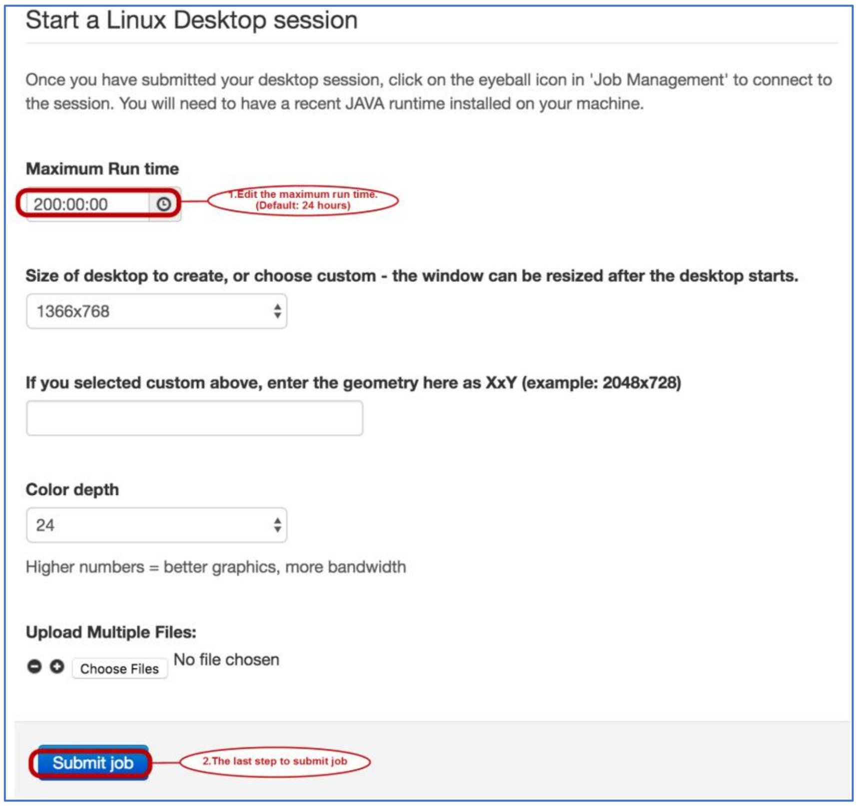
Submit a Job

Path: Forms

- BIO192
  Start a BIO192 Desktop session
- CSE427
  Start a CSE427 desktop session - LIMITED ACCESS
- Fluent
  Start a Fluent session
- Linux_Desktop
  Start a full desktop session for shell or abaqus
- Linux_Test
  Start a full desktop session for shell or abaqus
- Matlab
  Start a Matlab session

Figure 2.2 Selection Page of Submit job

In the desktop session page, you will need to set the maximum run time and then click the button “Submit job”(Figure 2.3).



Start a Linux Desktop session

Once you have submitted your desktop session, click on the eyeball icon in 'Job Management' to connect to the session. You will need to have a recent JAVA runtime installed on your machine.

Maximum Run time: 200:00:00

Size of desktop to create, or choose custom - the window can be resized after the desktop starts.: 1366x768

If you selected custom above, enter the geometry here as XxY (example: 2048x728)

Color depth: 24

Higher numbers = better graphics, more bandwidth

Upload Multiple Files: No file chosen

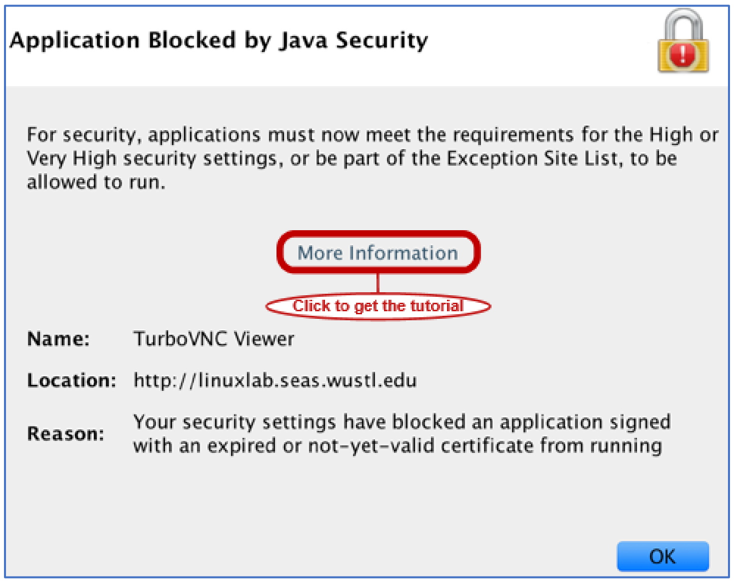
Figure 2.3 Submit job of Desktop

Remark: There will be errors when using “Upload Multiple Files” at present, so we will present another method to upload files onto our server.

After submit the job, you will download the file “xsession.jnlp”. The file need to be run by java. Therefore, if you cannot run it when you double click, go to the following website and download the latest “Java”.

- https://www.java.com/en/download/

After install "Java", double click "xsession.jnlp". You might get the information like figure 4. Then you need to add the "http://linuxlab.seas.wustl.edu" to the "Security Exception Set List" of "Java Control panel"(Click "more information" in Figure 2.4, you will know how to find "Java Control panel"). After add the exception site list(Figure 2.5), you can double click "xsession.jnlp" again and you will login in to the Linux Lab desktop(Figure 2.6).


Application Blocked by Java Security

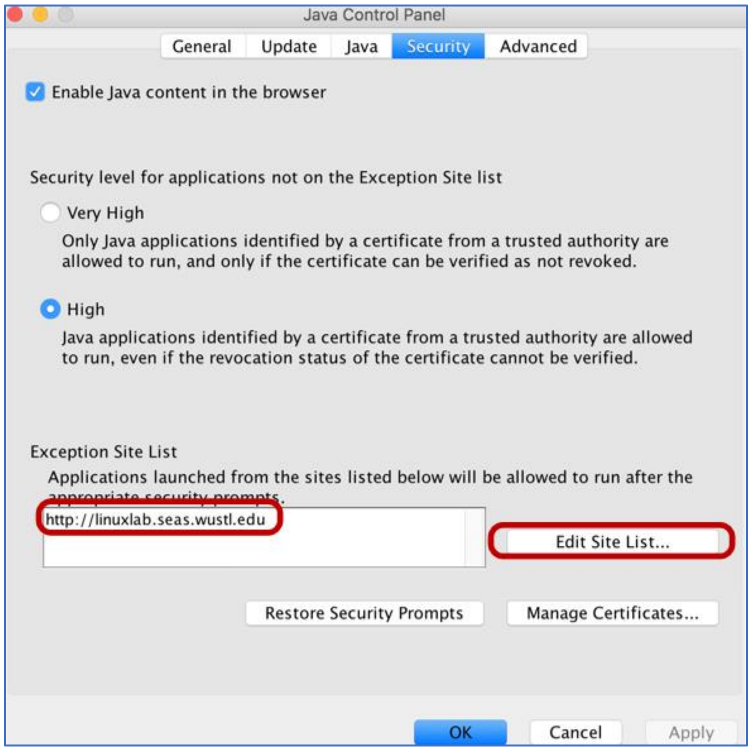
For security, applications must now meet the requirements for the High or Very High security settings, or be part of the Exception Site List, to be allowed to run.

Name: TurboVNC Viewer

Location: http://linuxlab.seas.wustl.edu

Reason: Your security settings have blocked an application signed with an expired or not-yet-valid certificate from running

Figure 2.4 Java Security



☑ Enable Java content in the browser

Security level for applications not on the Exception Site list

○ Very High
Only Java applications identified by a certificate from a trusted authority are allowed to run, and only if the certificate can be verified as not revoked.

● High
Java applications identified by a certificate from a trusted authority are allowed to run, even if the revocation status of the certificate cannot be verified.

Exception Site List
Applications launched from the sites listed below will be allowed to run after the appropriate security prompts.
http://linuxlab.seas.wustl.edu

Figure 2.5 Add Exception Site List

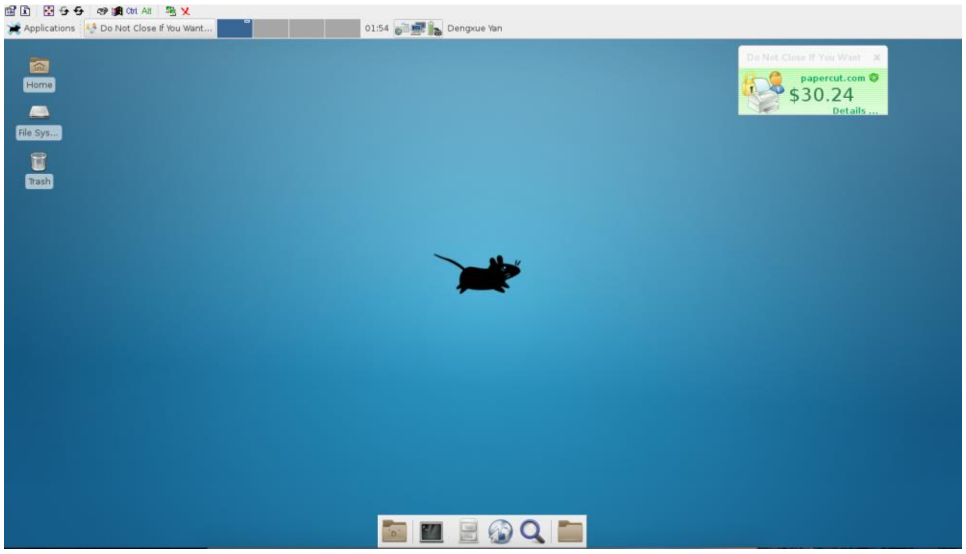


Figure 2.6 Linux Lab Desktop

2.2. Login to Linux Lab server using ssh

The Linux Lab server you can directly using either on-campus or off-campus is:

- shell.cec.wustl.edu

You could also using the following server on-campus or access them through **VPN** when off-campus(This tutorial will not provide detailed information about using VPN):

- linuxlab001.seas.wustl.edu
- linuxlab002.seas.wustl.edu
- linuxlab003.seas.wustl.edu
- linuxlab004.seas.wustl.edu
- linuxlab005.seas.wustl.edu
- linuxlab006.seas.wustl.edu
- linuxlab007.seas.wustl.edu
- linuxlab008.seas.wustl.edu
- linuxlab009.seas.wustl.edu
- linuxlab010.seas.wustl.edu
- linuxlab011.seas.wustl.edu
- linuxlab012.seas.wustl.edu

We could use ssh to login to these server.

2.2.1 Mac OS and Linux

In the terminal, enter the following command:

```
% ssh -AY UserName@SeverName
```

UserName is your Webstac ID and ServerName is a server from above list. For example:

```
% ssh -AY dengxue.yan@shell.cec.wustl.edu
```

Then you will be asked for password. Again enter your Webstac password and you will login to the Linux Lab server.

The parameter “AY” will allow you to run graphical programs of Linux Lab server on the local desktop without starting a remote desktop.

2.2.2 Windows

On windows, this is no “ssh” command by default, so we need to install programs to support it:

- SSH Secure Shell Client
- PuTTY

Remark: these two clients on windows cannot support graphical programs. (If you are interested, you can try it by yourself using Cygwin).

2.2.3 Load file to Linux Lab Server

Download filezilla from the following link:

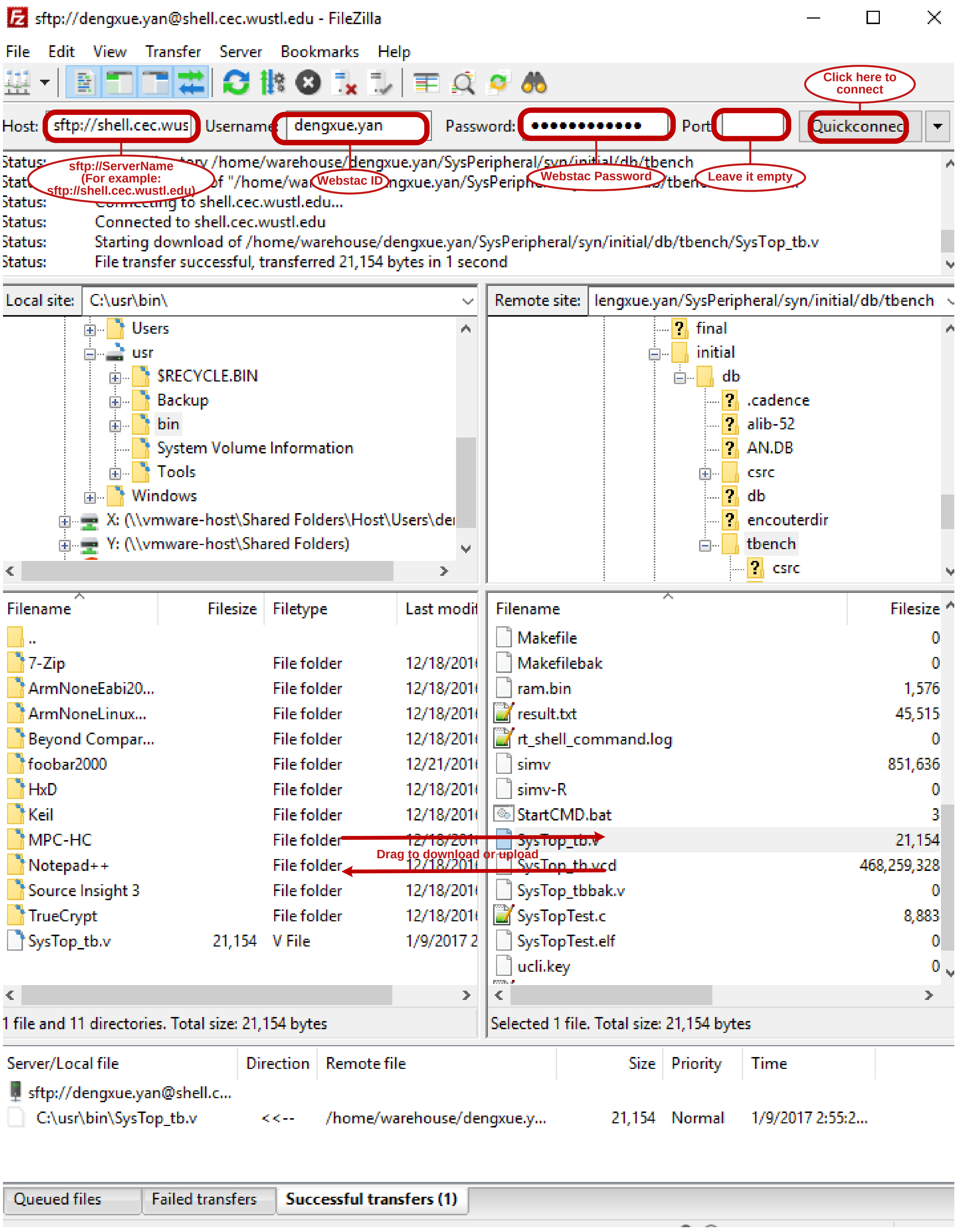
- https://filezilla-project.org/

After install it, fill the following items:

- Host: sttp://ServerName (e.g. sftp://shell.cec.wustl.edu)
- Username: Webstac ID
- Password: Webstac password

Then click “Quickconnect” to connect to the server.

After you see the files on the server, you can upload or download files by dragging them to the desired folder(Figure 2.7)


sftp://dengxue.yan@shell.cec.wustl.edu - FileZilla
File Edit View Transfer Server Bookmarks Help
Click here to connect
Host: sftp://shell.cec.wus
Username: dengxue.yan
Password: ●●●●●●●●●●●●
Port:
Quickconnect
sftp://ServerName (For example: sftp://shell.cec.wustl.edu)
Webstac ID
Webstac Password
Leave it empty
Drag to download or upload
Queued files
Failed transfers
Successful transfers (1)

```
Status:    ...tory /home/warehouse/dengxue.yan/SysPeripheral/syn/initial/db/tbench
Status:    ...of "/home/wa... ngxue.yan/SysPeriph... .../tbench
Status:    Connecting to shell.cec.wustl.edu...
Status:    Connected to shell.cec.wustl.edu
Status:    Starting download of /home/warehouse/dengxue.yan/SysPeripheral/syn/initial/db/tbench/SysTop_tb.v
Status:    File transfer successful, transferred 21,154 bytes in 1 second
```

Local site: C:\usr\bin\

```
Users
usr
    $RECYCLE.BIN
    Backup
    bin
    System Volume Information
    Tools
Windows
X: (\\vmware-host\Shared Folders\Host\Users\de
Y: (\\vmware-host\Shared Folders)
```

Remote site: lengxue.yan/SysPeripheral/syn/initial/db/tbench

```
final
initial
    db
        .cadence
        alib-52
        AN.DB
        csrc
        db
        encouterdir
        tbench
            csrc
```

Filename	Filesize	Filetype	Last modi
..			
7-Zip		File folder	12/18/201
ArmNoneEabi20...		File folder	12/18/201
ArmNoneLinux...		File folder	12/18/201
Beyond Compar...		File folder	12/18/201
foobar2000		File folder	12/21/201
HxD		File folder	12/18/201
Keil		File folder	12/18/201
MPC-HC		File folder	12/18/201
Notepad++		File folder	12/18/201
Source Insight 3		File folder	12/18/201
TrueCrypt		File folder	12/18/201
SysTop_tb.v	21,154	V File	1/9/2017 2

1 file and 11 directories. Total size: 21,154 bytes

Filename	Filesize
Makefile	0
Makefilebak	0
ram.bin	1,576
result.txt	45,515
rt_shell_command.log	0
simv	851,636
simv-R	0
StartCMD.bat	3
SysTop_tb.v	21,154
SysTop_tb.vcd	468,259,328
SysTop_tbbak.v	0
SysTopTest.c	8,883
SysTopTest.elf	0
ucli.key	0

Selected 1 file. Total size: 21,154 bytes

Server/Local file	Direction	Remote file	Size	Priority	Time
sftp://dengxue.yan@shell.c...					
C:\usr\bin\SysTop_tb.v	<<--	/home/warehouse/dengxue.y...	21,154	Normal	1/9/2017 2:55:2...

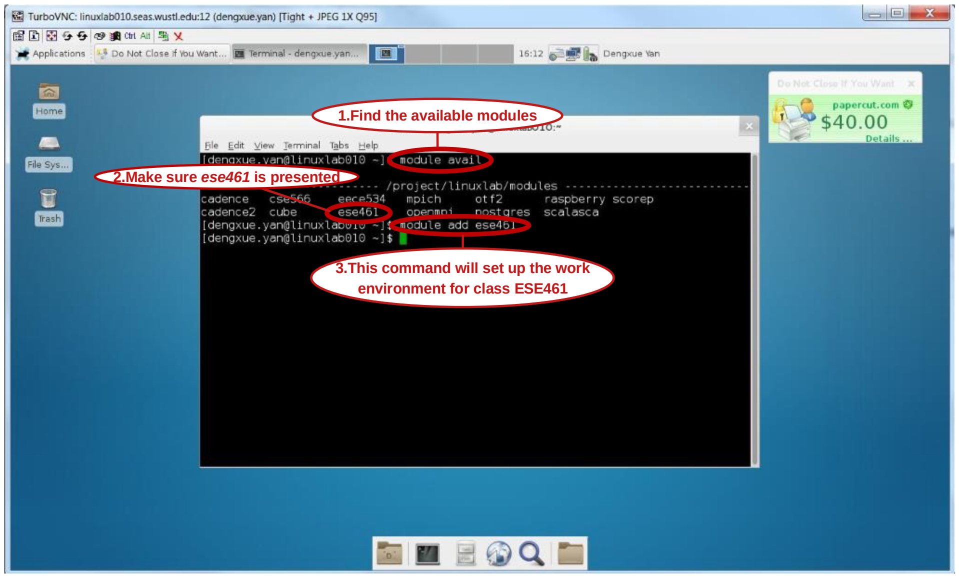
Figure 2.7 Filezilla

3. Build work environment for class ESE 566A

In the terminal, execute the following command:

% module add ese461

This command will build work environment for class ESE566A (also for ESE461). You could perform "*module avail*" in the terminal to find the available modules on Linuxlab before you do "*module add ese461*" .Make sure ese461 is presented when you execute this command.



```
TurboVNC: linuxlab010.seas.wustl.edu:12 (dengxue.yan) [Tight + JPEG 1X Q95]
[dengxue.yan@linuxlab010 ~]$ module avail
------------- /project/linuxlab/modules -------------------------
cadence   cse566   eece534   mpich     otf2      raspberry scorep
cadence2  cube     ese461    openmpi   postgres  scalasca
[dengxue.yan@linuxlab010 ~]$ module add ese461
[dengxue.yan@linuxlab010 ~]$
```

1.Find the available modules
2.Make sure *ese461* is presented
3.This command will set up the work environment for class ESE461
Applications
Do Not Close If You Want...
Terminal - dengxue.yan...
16:12
Dengxue Yan
Home
File Sys...
Trash
File Edit View Terminal Tabs Help
papercut.com
$40.00
Details ...

Figure 3.1 Build work environment for class ESE461 using module

4. The Linux Command Line[1]

The shell is the original Linux user interface which is a text-based command-line interpreter. The default shell on the Linux Lab is Bash. While there are other shells such as sh, csh, and tcsh, for this course we will always be assuming you are using Bash. As mentioned above, we use the % character to indicate commands that should be entered at the Linux command line, but you should not include the actual % character when typing in the commands on your own. To make it easier to cut-and-paste commands from this tutorial document onto the command line, you can tell Bash to ignore the '%' character using the following command: % alias %="" Now you can cut-and-paste a sequence of commands from this tutorial document and Bash will not get confused by the '%' character which begins each line.

4.1. Hello World

We begin with the ubiquitous "Hello, World" example. To display the message "Hello, World" we will use the echo command. The echo command simply "echo" its input to the console.

```
% echo "Hello, World"
```

The string we provide to the echo command is called a command line argument. We use command line arguments to tell commands what they should operate on. Although simple, the echo command can very useful for creating simple text files, displaying environment variables, and general debugging.

4.2. Manual Pages

You can learn more about any Linux command by using the man command. Try using this to learn more about the echo command.

```
% man echo
```

You can use the up/down keys to scroll the manual one line at a time, the space bar to scroll down one page at a time, and the q key to quit viewing the manual. You can even learn about the man command itself by using man man. As you follow the tutorial, feel free to use the man command to learn more about the commands we cover.

4.3. Create, View, and List Files

We can use the echo command and a feature called *command output* redirection to create simple text files. We will discuss command output redirection in more detail later in the tutorial. Command output redirection uses the “>” operator to take the output from one command and “redirect” it to a file. The following commands will create a new file named ese566a-tutorial1.txt that simply contains the text “Modern System-on-Chip Design”.

```
% echo "Modern System-on-Chip Design" > ese566a-tutorial1.txt
```

We can use the “cat” command to quickly display the contents of a file.

```
% cat ese566a-tutorial1.txt
```

For larger files, cat will output the entire file to the console so it may be hard to read the file as it streams past. We can use the less command to show one screen-full of text at a time. You can use the up/down keys to scroll the file one line at a time, the space bar to scroll down one page at a time, and the q key to quit viewing the file.

```
% less ese566a-tutorial1.txt
```

You can use the “ls” command to list the filenames of the files you have created.

```
% ls
```

We can provide command line options to the “ls” command to modify the command’s behavior. For example, we can use the -1 (i.e., a dash followed by the number one) command line option to list one file per line, and we can we can use the -l (i.e., a dash followed by the letter l) command line option to provide a longer listing with more information about each file.

```
% ls -1
% ls -l
```

You should see the newly created ese566a-tutorial1.txt file along with some additional directories or folders. We will discuss directories in the next section. Use the following commands to create a few more files using the echo command and command output redirection, and then list the files again.

```
% echo "Application" > ese566a-tutorial1-layer1.txt
% echo "Algorithm"   > ese566a-tutorial1-layer2.txt
% ls -1
```

4.4. Create, Change, and List Directories

Obviously, having all files in a single location would be hard to manage effectively. We can use directories (also called folders) to logically organize our files, just like one can use physical folders to organize physical pieces of paper. The mechanism for organizing files and directories is called the file system. When you first login to an Linux Lab server, you will be in your *home directory*. This is your own private space on the server that you can use to work on the lab assignments and store your files. You can use the pwd command to print the directory in which you are currently working, which is known as the *current working directory*.

```
% pwd
/home/warehouse/<UserName>
```

You should see output similar to what is shown above, but instead of <UserName> it should show your actual Webstac ID. The pwd command shows a directory path. A directory path is a list of nested directory names; it describes a “path” to get to a specific file or directory. So the above path indicates that there is a toplevel directory named home that contains a directory named <UserName>. This is the directory path to your home directory. As an aside, notice that Linux uses a forward slash (/) to separate directories, while Windows uses a back slash (\) for the same purpose.

We can use the mkdir command to make new directories. The following command will make a new directory named ese566A within your home directory.

```
% mkdir ese566A
```

We can use the cd command to change our current working directory. The following command will change the current working directory to be the newly created ese566A directory, before displaying the current working directory with the pwd command.

```
% cd ese566A
% pwd
/home/warehouse/<UserName>/ese566A
```

Use the mkdir, cd, and pwd commands to make another directory.

```
% mkdir LinuxTutorial
% cd LinuxTutorial
% pwd
/home/warehouse/<UserName>/ese566A/LinuxTutorial
```

We sometimes say that *LinuxTutorial* is a subdirectory or a child directory of the *ese566A* directory. We might also say that the *ese566A* directory is the parent directory of the *LinuxTutorial* directory.

There are some important shortcuts that we can use with the cd command to simplify navigating the file system. The special directory named . (i.e., one dot) always refers to the current working directory. The special directory named .. (i.e., two dots) always refers to the parent of the current working directory. The special directory named ~ (i.e., a tilde character) always refers to your home directory. The special directory named / (e.g., single forward slash) always refers to the highest level root directory.

The following commands create a few more directories before displaying the directory hierarchy.

```
% cd ~/ese566A/LinuxTutorial
% mkdir -p dirB/dirB_1
% mkdir -p dirB/dirB_2
% mkdir -p dirC/dirC_1
% cd ~/ese566A/LinuxTutorial
% cd ~
```

Note that we are using the -p command line option with the mkdir command to make multiple nested directories in a single step. And the *cd* command with no command line arguments always changes the current working directory to your home directory. We can use the ls command to list files as well as directories. Use the following commands to create a new file and directory in the *ese566A/LinuxTutorial* subdirectory, and then list the file and directory.

```
% cd ~/ese566A/LinuxTutorial
% echo "Computer Architecture" > ese566a-tutorial1.txt
% mkdir dirA
% ls -1
```

4.5. Copy, Move, and Remove Files and Directories

We can use the cp command to copy files. The first argument is the name of the file you want to copy, and the second argument is the new name to give to the copy. The following commands will make two copies of the files we created in the previous section.

```
% cd ~/ese566A/LinuxTutorial
% cp ese566a-tutorial1.txt ese566a-tutorial1-a.txt
% cp ese566a-tutorial1.txt ese566a-tutorial1-b.txt
% ls -1
```

We can also copy one or more files into a subdirectory by using multiple source files and a final destination directory as the arguments to the cp command.

```
% cd ~/ese566A/LinuxTutorial
% cp ese566a-tutorial1.txt dirA
% cp ese566a-tutorial1-a.txt ese566a-tutorial1-b.txt dirB
```

We can use the -r command line option to enable the cp command to recursively copy an entire directory.

```
% cd ~/ese566A/LinuxTutorial
% cp -r dirA dirD
```

If we want to move a file or directory, we can use the mv command. As with the cp command, the first argument is the name of the file you want to move and the second argument is the new name of the file.

```
% cd ~/ese566A/LinuxTutorial
% mv ese566a-tutorial1.txt ese566a-tutorial1-c.txt
% ls -1
```

Again, similar to the cp command, we can also move one or more files into a subdirectory by using multiple source files and a final destination directory as the arguments to the mv command.

```
% cd ~/ese566A/LinuxTutorial
% mv ese566a-tutorial1-a.txt dirB
% mv ese566a-tutorial1-b.txt ese566a-tutorial1-c.txt dirB
```

We do not need to use the -r command line option to move an entire directory at once.

```
% cd ~/ese566A/LinuxTutorial/
% ls -1
% mv dirD dirE
% ls -1
```

The following example illustrates how we can use the special . directory to move files from a subdirectory into the current working directory.

```
% cd ~/ese566A/LinuxTutorial/
% ls -1
% mv dirE/ese566a-tutorial1.txt .
% ls -1
```

We can use the “rm” command to remove files. The following command removes a file from within the ese566A/LinuxTutorial subdirectory.

```
% cd ~/ese566A/LinuxTutorial/
% ls -1
% rm ese566a-tutorial1.txt
% ls -1
```

To clean up, we might want to remove the files we created in your home directory earlier in this tutorial.

```
% cd
% rm ese566a-tutorial1.txt
% rm ese566a-tutorial1-layer1.txt
% rm ese566a-tutorial1-layer2.txt
```

We can use the -r command line option with the rm command to remove entire directories, but please be careful because it is relatively easy to permanently delete many files at once.

```
% cd ~/ese566A/LinuxTutorial
% ls -1
% rm -r dirA dirB dirC dirE
% ls -1
```

4.6. Using wget to Download Files

We can use the wget command to download files from the internet. For now, this is a useful way t retrieve a text file that we can use in the following examples.

```
% cd ~/ese566A/LinuxTutorial
% wget http://classes.engineering.wustl.edu/ese566/overview.txt
% cat overview.txt
```

4.7. Using grep to Search Files

We can use the grep command to search and display lines of a file that contain a particular pattern. The grep command can be useful for quickly searching the contents of the source files in your lab assignment. The command takes the pattern and the files to search as command line arguments. The following command searches for the word “memories” in the overview.txt file downloaded in the previous section.

```
% cd ~/ese566A/LinuxTutorial
% grep "system" overview.txt
```

We can use the --line-number and --color command line options with the grep command to display the line number of each match and to highlight the matched word.

```
% cd ~/ese566A/LinuxTutorial
% grep --line-number --color "system" overview.txt
```

We can use the -r command line option to recursively search all files within a given directory hierarchy. In the following example, we create a subdirectory, copy the overview.txt file, and illustrate how we can use the grep command to recursively search for the word “system”.

```
% cd ~/ese566A/LinuxTutorial
% mkdir dirA
% cp overview.txt dirA
% grep -r --line-number --color "system" .
```

Notice how we specify a directory as a command line argument (in this case the special . directory) to search the current working directory. You should see the three lines from both copies of the overview.txt file. The grep command also shows which file contains the match.

4.8. Using find to Find Files

We can use the find command to recursively search a directory hierarchy for files or directories that match a specified criteria. While the grep command is useful for searching file contents, the find command is useful for quickly searching the file and directory names in your lab assignments. The find command is very powerful, so we will just show a very simple example. First, we create a few new files and directories.

```
% cd ~/ese566A/LinuxTutorial
% mkdir -p dirB/dirB_1
% mkdir -p dirB/dirB_2
% mkdir -p dirC/dirC_1
% echo "test" > dirA/file0.txt
% echo "test" > dirA/file1.txt
% echo "test" > dirB/dirB_1/file0.txt
% echo "test" > dirB/dirB_1/file1.txt
% echo "test" > dirB/dirB_2/file0.txt
```

We will now use the find command to find all files named "file0.txt". The find command takes one command line argument to specify where we should search and a series of command line options to describe what files and directories we are trying to find. We can also use command line options to describe what action we would like to take when we find the desired files and directories. In this example, we use the —name command line option to specify that we are searching for files with a specific name. We can also use more complicated patterns to search for all files with a specific filename prefix or extension.

```
% cd ~/ese566A/LinuxTutorial
% find . -name "file0.txt"
```

Notice that we are using the special . directory to tell the find command to search the current working directory and all subdirectories. The find command always searches recursively

4.9. Using tar to Archive Files

We can use the tar command to “pack” files and directories into a simple compressed archive, and also to “unpack” these files and directories from the archive. This kind of archive is sometimes called a tarball. Most open-source software is distributed in this compressed form. It makes it easy to distribute code among collaborators and it is also useful to create backups of files. We can use the following command to create an archive of our tutorial directory and then remove the tutorial directory.

```
% cd ~/ese566A
% tar -czvf LinuxTutorial.tgz LinuxTutorial
% rm -r LinuxTutorial
% ls -l
```

Several command line options listed together as a single option (-czvf), where c specifies we want to create an archive, z specifies we should use “gzip” compression, v specifies verbose mode, and f specifies we will provide filenames to archive. The first command line argument is the name of the archive to create, and the second command line argument is the directory to archive. We can now extract the contents of the archive to recreate the tutorial directory. We also remove the archive.

```
% cd ~/ese566A
% tar -xzvf LinuxTutorial.tgz
% rm LinuxTutorial.tgz
% ls -l
```

Note that we use the x command line option with the tar command to specify that we intend to extract the archive.

4.10. Using top to View Running Processes

You can use the top command to view what commands are currently running on the Linux system in realtime. This can be useful to see if there are many commands running which are causing the system to be sluggish. When finished you can use the q character to quit.

```
% top
```

The first line of the top display shows the number of users currently logged into the system, and the load average. The load average indicates how “overloaded” the system was over the last one, five, and 15 minutes. If the load average is greater than the number of processors in the system, it means your system will probably be sluggish. You can always try logging into a different server in the cluster.

4.11. Environment Variables

In the previous sections, we have been using the Bash shell to run various commands, but the Bash shell is actually a full-featured programming language. One aspect of the shell that is similar in spirit to popular programming languages, is the ability to write and read environment variables. The following commands illustrate how to write an environment variable named ese566A_tut1_layer1, and how to read this environment variable using the echo command.

```
% ese566A_tut1_layer1="application"
% echo ${ese566A_tut1_layer1}
```

Keep in mind that the names of environment variables can only contain letters, numbers, and underscores. Notice how we use the ${} syntax to read an environment variable. There are a few built-in environment variables that might be useful:

```
% echo ${HOSTNAME}
% echo ${HOME}
% echo ${PWD}
```

We often use the HOME environment variable in directory paths like this:

```
% cd ${HOME}/ese566A
```

The PWD environment variable always holds the current working directory. We can use environment variables as options to commands other than echo. A common example is to use an environment variable to "remember" a specific directory location, which we can quickly return to with the cd command like this:

```
% cd ${HOME}/ese566A/LinuxTutorial
% LinuxTutorial=${PWD}
% cd
% pwd
/home/warehare/<UserName>
% cd ${LinuxTutorial}
% pwd
/home/warehouse/<UserName>/ese566A/LinuxTutorial
```

4.12. Command Output Redirection

We have already seen using the echo command and command output redirection to create simple text files. Here is another example:

```
% cd ${HOME}/ese566A/LinuxTutorial
% echo "Application" > computing-stack.txt
% cat computing-stack.txt
```

The > operator tells the Bash shell to take the output from the command on the left and overwrite the file named on the right. We can use any command on the left. For example, we can save the output from the pwd command or the man command to a file for future reference.

```
% cd ${HOME}/ese566A/LinuxTutorial
% pwd > cmd-output.txt
% cat cmd-output.txt
% man pwd > cmd-output.txt
% cat cmd-output.txt
```

We can also use the >> operator which tells the Bash shell to take the output from the command on the left and append the file named on the right. We can use this to create multiline text files:

```
% cd ${HOME}/ese566A/LinuxTutorial
% echo "Algorithm" > computing-stack.txt
% echo "Programming Language" >> computing-stack.txt
% echo "Operating System" >> computing-stack.txt
% cat computing-stack.txt
```

4.13. Command Chaining

We can use the && operator to specify two commands that we want to chaining together. The second command will only execute if the first command succeeds. Below is an example.

```
% cd ${HOME}/ese566A/LinuxTutorial && cat computing-stack.txt
```

4.14. Command Pipelining

The Bash shell allows you to run multiple commands simultaneously, with the output of one command becoming the input to the next command. We can use this to assemble “pipelines”; we “pipe” the output of one command to another command for further actions using the | operator. The following example uses the grep command to search the special proc/cpuinfo file for lines containing the word “processor” and then pipes the result to the wc command. The wc command counts the number of characters, words, or lines of its input. We use the -l command line option with the wc command to count the number of lines.

```
% grep processor /proc/cpuinfo | wc -l
```

This is a great example of the Linux philosophy of providing many simple commands that can be combined to create more powerful functionality. Essentially the pipeline we have created is a command that tells us the number of processors in our system.

As another example, we will pipe the output of the *grep aux* command to the grep command. The *grep aux* command report a snapshot of all the current processes. We can use grep to search for your Webstac ID and thus quickly see what you are running on the server.

```
% ps aux | grep <UserName>
```

We can create even longer pipelines. The following pipeline will report the number of processes you are currently running on the server.

```
% ps aux | grep <UserName> | wc -l
```

4.15. Wildcards and Tab Completion

In this section, we describe some miscellaneous features of the Bash shell which can potentially be quite useful in increasing your productivity.

Wildcards make it easy to manipulate many files and directories at once. Whenever we specify a file or directory on the command line, we can often use a wildcard instead. In a wildcard, the asterisk (*) will match any sequence of characters. The following example illustrates how to list all files that end in the suffix .txt and then copies all files that match the wildcard from one directory to another.

```
% cd ${HOME}/ese566A/LinuxTutorial
% ls *.txt
% cp dirA/file*.txt dirB
```

The Bash shell supports tab completion. When you press the tab key twice after entering the beginning of a filename or directory name, Bash will try to automatically complete the filename or directory name. If there is more than one match, Bash will show you all of these matches so you can continue narrowing your search.

5. Conclusion

This tutorial hopefully helped you become familiar with Linux and how to use it for working on the labs. You have gained some experience working at the command line. There are many more resources online for learning Linux, and keep in mind that learning to work productively using the Linux operating system can pay dividends in many other contexts besides this course.

6. Reference

[1] http://www.csl.cornell.edu/courses/ece4750/handouts/ece4750-tut1-linux.pdf