ECE 4750 Computer Architecture, Fall 2024
Tutorial 0: Remote Access to `ecelinux`

School of Electrical and Computer Engineering
Cornell University

revision: 2024-08-28-22-12

1. Introduction

All of the laboratory assignments for this course will be completed by remotely logging into a cluster of `ecelinux` servers. The `ecelinux` servers all run the Red Hat Enterprise Linux 7 operating system, and they all use an identical setup. You do not need to do anything special to create an `ecelinux` account. You will be using your NetID and Cornell password to login, and an `ecelinux` account will be automatically created for you when you first login. Any student enrolled in any ECE class should automatically be granted access to the `ecelinux` servers. Having said this, if you cannot log into the `ecelinux` servers please reach out to the course staff for assistance.

Later tutorials will discuss how to use the Linux development environment and the Git distributed version control system. In this tutorial, we focus on how to setup remote access to the `ecelinux` servers by first connecting to the Cornell VPN and then choosing one of several different remote access options. Which remote access option to use depends on your operating system, whether or not you are planning to use Linux applications with a graphical user interface (GUI), the speed of your network connection, and your own personal preference. The following table provides a summary of these remote access options.

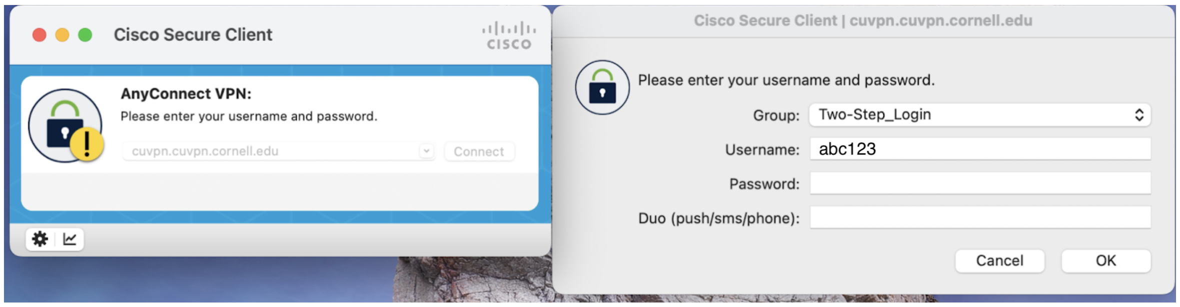
Remote Access Option	**Operating System**	**Can use Linux command line?**	**Can use Linux applications with a GUI?**	**Internet Speed Requirements**
Section 4: PowerShell	Windows	Yes	No	Low
Section 5: Mac Terminal	Mac OS X	Yes	No	Low
Section 6: VS Code	either	Yes	No	Very Low
Section 7: X2Go	either	Yes	Yes	Medium
Section 8: MobaXterm	Windows	Yes	Yes	High
Section 9: Mac Terminal w/ XQuartz	Mac OS X	Yes	Yes	High

We recommend at least trying to use PowerShell or the Mac Terminal as your first attempt at remote access to ensure your connection is working. Using PowerShell or the Mac Terminal can be quite primitive though, so we recommend primarily using these as backup options throughout the course in case you are having trouble with the other options. **We strongly recommend using VS Code as your primary remote access option for code development, and then using X2Go only if you need to use a Linux application with a GUI.** MobaXterm or Mac Terminal with XQuartz are also perfectly fine options, but they are only feasible if you are on a relatively high-speed internet connection to the `ecelinux` servers (i.e., on the Campus network).

2. Connecting to the Cornell VPN

If you are logging into the `ecelinux` servers from on campus (i.e., using the Cornell wired or wireless network), then you do not need to enable the Cornell virtual private network (VPN). However, if you are off campus, then you will need to enable the Cornell VPN whenever you want to log into the `ecelinux` servers. The VPN provides very secure access to all on-campus network resources. More information about the Cornell VPN is available here:

- `https://it.cornell.edu/cuvpn`


Cisco Secure Client
AnyConnect VPN:
Please enter your username and password.
cuvpn.cuvpn.cornell.edu
Connect
Cisco Secure Client | cuvpn.cuvpn.cornell.edu
Please enter your username and password.
Cancel
OK

Group: Two-Step_Login
Username: abc123
Password:
Duo (push/sms/phone):

Figure 1: Running the Cisco AnyConnect Client for Cornell VPN

Simply follow the instructions at the following link to install the Cisco VPN software for the appropriate operating system you use on your laptop/workstation:

- `https://it.cornell.edu/landing-page-kba/2605/5273`

Once the Cornell VPN is installed, then connect to the Cornell VPN by following these instructions and using your Cornell NetID and password:

- `https://it.cornell.edu/landing-page-kba/2605/823`

The Cornell VPN uses the Cisco AnyConnect Client, and Figure 1 illustrates running the client.

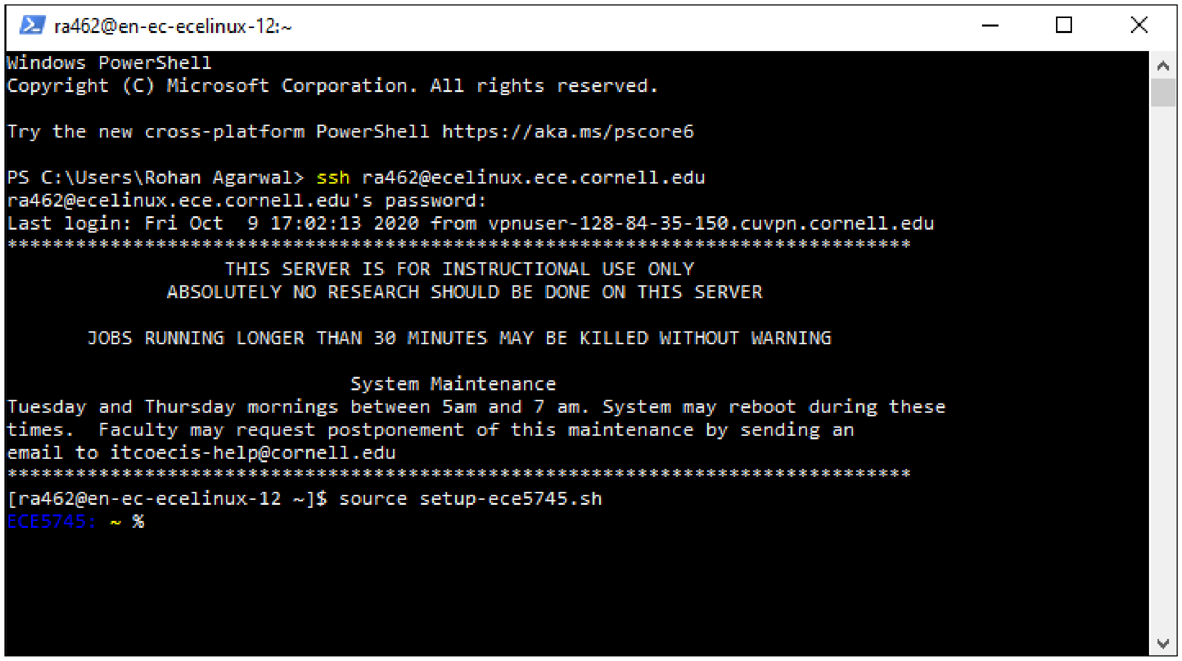
3. Heads up about `ecelinux`

This tutorial repeatedly tells you to log onto a machine called `ecelinux.ece.cornell.edu`. In fact, `ecelinux.ece.cornell.edu` is not a machine that you log into but rather a load-balancer that will log you on to one of a few possible `ecelinux` machines. When you `ssh` into `ecelinux.ece.cornell.edu`, you may actually be sent to `ecelinux-01.ece.cornell.edu` or `ecelinux-02.ece.cornell.edu`, for example. This is like when everyone at a store waits in one long line for a cashier and then when you're first in line, you go to whichever cashier happens to be free. `ecelinux.ece.cornell.edu` is the long line that everyone waits in, and `ecelinux-02.ece.cornell.edu` might be the machine that has the best available resources for you at the time.

That's all well and good, but sometimes, the load-balancer does not do its job. The department is in the process of trying to improve it. In the meantime, throughout this tutorial, if you ever are unable to log onto `ecelinux.ece.cornell.edu` (after making sure that you are on the VPN), we recommend that you try to go straight to the source. Try ssh-ing directly onto `ecelinux-01.ece.cornell.edu` or `ecelinux-02.ece.cornell.edu` by just using those names directly wherever you were previously using `ecelinux.ece.cornell.edu`.

4. Remote Access via PowerShell

PowerShell is part of Windows OS, and it enables interacting with your Windows OS system from the command line (i.e., a powerful text-based environment where users type commands to manipulate files and directories and execute applications). PowerShell also enables remotely accessing other systems (e.g., the `ecelinux` servers) via the command line using SSH, a highly secure network protocol and associated client/server program. PowerShell will enable you to log into the `ecelinux` servers and to then manipulate files and directories and execute applications remotely on the `ecelinux` servers using the Linux command line. Note that PowerShell does not enable using



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Rohan Agarwal> ssh ra462@ecelinux.ece.cornell.edu
ra462@ecelinux.ece.cornell.edu's password:
Last login: Fri Oct  9 17:02:13 2020 from vpnuser-128-84-35-150.cuvpn.cornell.edu
******************************************************************************
                   THIS SERVER IS FOR INSTRUCTIONAL USE ONLY
              ABSOLUTELY NO RESEARCH SHOULD BE DONE ON THIS SERVER

       JOBS RUNNING LONGER THAN 30 MINUTES MAY BE KILLED WITHOUT WARNING

                              System Maintenance
Tuesday and Thursday mornings between 5am and 7 am. System may reboot during these
times.  Faculty may request postponement of this maintenance by sending an
email to itcoecis-help@cornell.edu
******************************************************************************
[ra462@en-ec-ecelinux-12 ~]$ source setup-ece5745.sh
ECE5745: ~ %
```

Figure 2: PowerShell After Logging Into ecelinux Servers

Linux applications with a GUI. If you need to use a Linux application with a GUI, you will need to choose a different remote access option (e.g., MobaXterm described in Section 8).

4.1. Starting and Configuring PowerShell

First, if you are off campus, then you must be connected to the Cornell VPN before attempting to access the ecelinux servers (see Section 2). To start PowerShell click the *Start* menu then choose *All Programs* > *Accessories* > *Windows PowerShell* > *Windows PowerShell*, or click the *Start* menu, type *PowerShell*, and choose *Windows PowerShell*.

The default blue background for PowerShell can be a little annoying. You can change the background to black or white by right clicking on the PowerShell window's title bar and then choose *Properties* > *Colors*, select *Screen Background*, click the black or white color, and click *OK*.

4.2. Logging into ecelinux Servers with PowerShell

After starting PowerShell, type in the following command at the prompt to log into the ecelinux servers using SSH.

```
% ssh netid@ecelinux.ece.cornell.edu
```

Replace netid with your Cornell NetID in the command above. You should not enter the % character. We use the % character to indicate what commands we should enter on the command line. Executing the command will prompt you to enter your Cornell NetID password, and then you should be connected to the ecelinux servers.

The very first time you log into the ecelinux servers you may see a warning like this:

```
The authenticity of host 'ecelinux.ece.cornell.edu (128.253.51.206)' can't be established.
```

```
ECDSA key fingerprint is SHA256:smwMnf9dyhs5zW5I279C5oJBrTFc5FLghIJMfBR1cxI.
Are you sure you want to continue connecting (yes/no)?
```

The very first time you log into the `ecelinux` servers it is okay to enter *yes*, but from then on if you continue to receive this warning please contact the course staff.

Once you have opened a terminal, the very first thing you need to do after logging into the `ecelinux` servers is source the course setup script. This will ensure your environment is setup with everything you need for working on the laboratory assignments. Enter the following command on the command line:

```
% source setup-ece4750.sh
```

Again, you should not enter the `%` character. You should now see a blue `ECE 4750` in your prompt which means your environment is setup for the course. Figure 2 shows what your prompt should look like if you have sourced the course setup script. See Section 11 for more on how to automatically source the setup script every time you log into the `ecelinux` servers, and see Section 10 for more on text editors you can use in the course.

5. Remote Access via Mac Terminal

Mac Terminal is part of Mac OS X, and it enables interacting with your Mac OS X system from the command line (i.e., a powerful text-based environment where users type commands to manipulate files and directories and execute applications). Mac Terminal also enables remotely accessing other systems (e.g., the `ecelinux` servers) via the command line using SSH, a highly secure network protocol and associated client/server program. Mac Terminal will enable you to log into the `ecelinux` servers and to then manipulate files and directories and execute programs remotely on the `ecelinux` servers using the Linux command line. Note that Mac Terminal (by default) does not enable using Linux applications with a GUI. If you need to use a Linux application with a GUI, you will need to choose a different remote access option (e.g., Mac Terminal with XQuartz described in Section 9).

5.1. Starting and Configuring Mac Terminal

First, if you are off campus, then you must be connected to the Cornell VPN before attempting to access the `ecelinux` servers (see Section 2). To start Mac Terminal go to your *Applications* folder and choose *Utilities* > *Terminal.app*, or open Spotlight (cmd + space), type *Terminal*, and press enter.

5.2. Logging into `ecelinux` Servers with Mac Terminal

After starting Mac Terminal, type in the following command at the prompt to log into the `ecelinux` servers using SSH.

```
% ssh netid@ecelinux.ece.cornell.edu
```

Replace `netid` with your Cornell NetID in the command above. You should not enter the `%` character. We use the `%` character to indicate what commands we should enter on the command line. Executing the command will prompt you to enter your Cornell NetID password, and then you should be connected to the `ecelinux` servers.

The very first time you log into the `ecelinux` servers you may see a warning like this:

```
The authenticity of host 'ecelinux.ece.cornell.edu (128.253.51.206)' can't be established.
```

```
ECDSA key fingerprint is SHA256:smwMnf9dyhs5zW5I279C5oJBrTFc5FLghIJMfBR1cxI.
Are you sure you want to continue connecting (yes/no)?
```

The very first time you log into the `ecelinux` servers it is okay to enter *yes*, but from then on if you continue to receive this warning please contact the course staff.

Once you have opened a terminal, the very first thing you need to do after logging into the `ecelinux` servers is source the course setup script. This will ensure your environment is setup with everything you need for working on the laboratory assignments. Enter the following command on the command line:

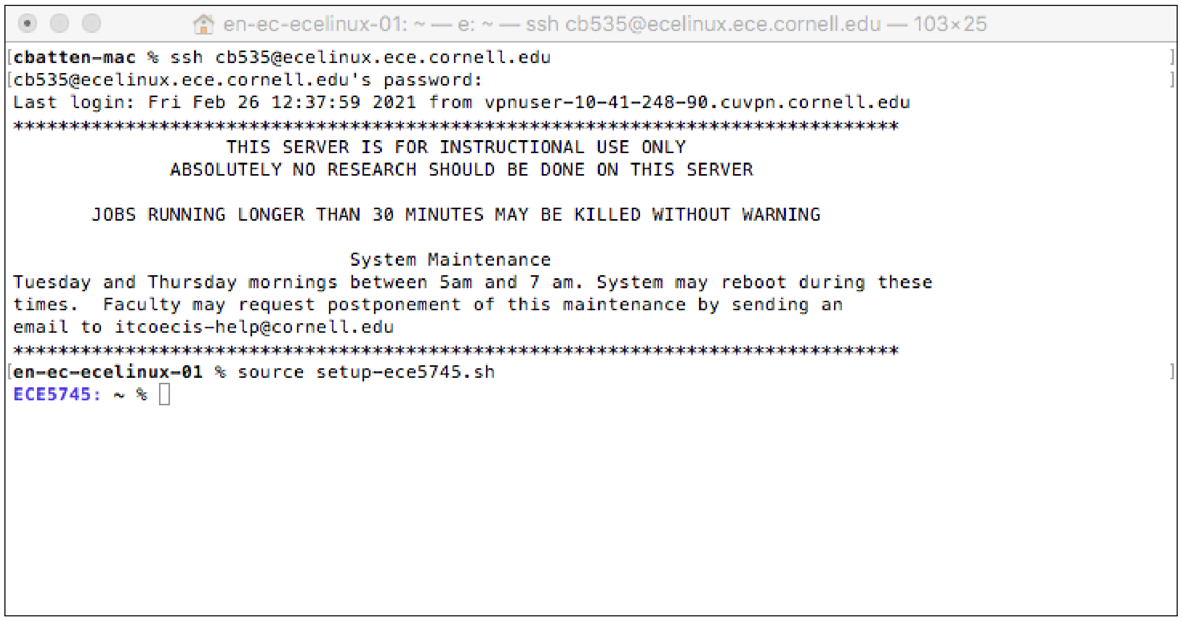
```
% source setup-ece4750.sh
```

Again, you should not enter the `%` character. You should now see a blue `ECE 4750` in your prompt which means your environment is setup for the course. Figure 3 shows what your prompt should look like if you have sourced the course setup script. See Section 11 for more on how to automatically source the setup script every time you log into the `ecelinux` servers, and see Section 10 for more on text editors you can use in the course.

6. Remote Access via VS Code

While combining PowerShell (see Section 4) or Mac Terminal (see Section 5) with a text-based editor such as nano can certainly work, it is not the most productive development setup. We strongly recommend using VS Code as your primary remote access option for code development (this Section). VS Code offers a nice balance of productive features while also working well with moderate internet speeds.

VS Code uses a unique approach where the GUI interface runs completely on your local laptop/-workshop and then automatically handles copying files back and forth between your local laptop/-


en-ec-ecelinux-01: ~ — e: ~ — ssh cb535@ecelinux.ece.cornell.edu — 103×25

```
cbatten-mac % ssh cb535@ecelinux.ece.cornell.edu
cb535@ecelinux.ece.cornell.edu's password:
Last login: Fri Feb 26 12:37:59 2021 from vpnuser-10-41-248-90.cuvpn.cornell.edu
*******************************************************************************
                   THIS SERVER IS FOR INSTRUCTIONAL USE ONLY
              ABSOLUTELY NO RESEARCH SHOULD BE DONE ON THIS SERVER

       JOBS RUNNING LONGER THAN 30 MINUTES MAY BE KILLED WITHOUT WARNING

                              System Maintenance
Tuesday and Thursday mornings between 5am and 7 am. System may reboot during these
times.  Faculty may request postponement of this maintenance by sending an
email to itcoecis-help@cornell.edu
*******************************************************************************
en-ec-ecelinux-01 % source setup-ece5745.sh
ECE5745: ~ %
```

Figure 3: Mac Terminal After Logging Into `ecelinux` Servers

workshop and the `ecelinux` servers. VS Code is portable across many operating systems and has a thriving ecosystem of extensions and plugins enabling it to function as a full-featured IDE for languages from C to Javascript. More information about VS Code is here:

- `https://code.visualstudio.com`
- `https://code.visualstudio.com/docs`

6.1. Installing VS Code on Your Laptop/Workstation

You can download VS Code by simply going to the main VS Code webpage:

- `https://code.visualstudio.com`

There should be an obvious link that says "Download for Windows" or "Download for Mac". Click on that link. On Mac OS X, you will need to drag the corresponding *Visual Student Code.app* to your *Applications* folder.

6.2. Starting and Configuring VS Code

First, if you are off campus, then you must be connected to the Cornell VPN before attempting to access the `ecelinux` servers (see Section 2). Start by opening VS Code. The exact way you do this will depend on whether you are using a Windows or Mac OS X laptop/workstation.

The key to VS Code is installing the correct extensions. You will probably need extensions for the C/C++, Python, and Verilog languages in this course. We also want to install a special extension which will enable remotely accessing the `ecelinux` servers using SSH. Choose *View* > *Extensions* from the menubar. Enter the name of the extension in the "Search Extensions in Marketplace" and then click the blue *Install* button. Here are the names of the four extensions:

- C/C++
- Python
- Verilog-HDL/SystemVerilog/Bluespec SystemVerilog
- Remote - SSH

For the C/C++, Python, and Remote - SSH extensions it is critical to choose the extension developed by Microsoft. You should see the name Microsoft in the extension description. Figure 4 illustrates how to install all four extensions.

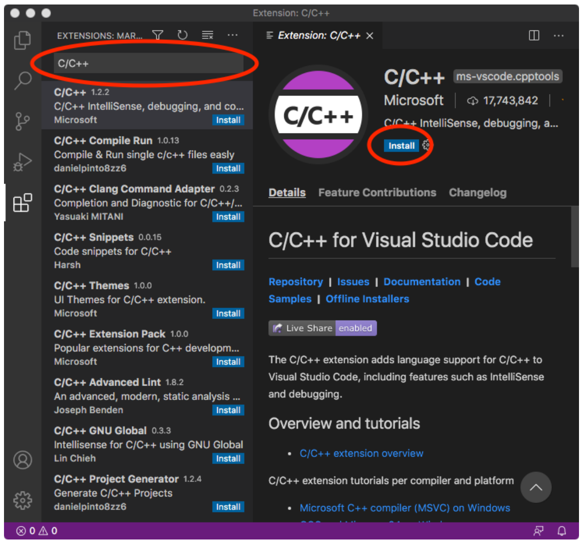
Now we need to setup the connection between VS Code which will run on your local laptop/workstation and the `ecelinux` servers. Choose *View* > *Command Palette* from the menubar. This will cause a little "command palette" to drop down where you can enter commands to control VS Code. Enter the following command in the command palette:

```
Remote-SSH: Add new SSH host...
```

As you start typing matching commands will be displayed and you can just click the command when you see it. VS Code will then ask you to *Enter SSH Connection Command*, and you should enter the following:

```
ssh netid@ecelinux.ece.cornell.edu
```

Replace `netid` with your Cornell NetID in the command above. VS Code will then ask you to *Select SSH configuration file to update*, and you should something like this on Windows:



C/C++ 1.2.2
C/C++ IntelliSense, debugging, and co...
Microsoft

C/C++ Compile Run 1.0.13
Compile & Run single c/c++ files easly
danielpinto8zz6

C/C++ Clang Command Adapter 0.2.3
Completion and Diagnostic for C/C++/...
Yasuaki MITANI

C/C++ Snippets 0.0.15
Code snippets for C/C++
Harsh

C/C++ Themes 1.0.0
UI Themes for C/C++ extension.
Microsoft

C/C++ Extension Pack 1.0.0
Popular extensions for C++ developm...
Microsoft

C/C++ Advanced Lint 1.8.2
An advanced, modern, static analysis ...
Joseph Benden

C/C++ GNU Global 0.3.3
Intellisense for C/C++ using GNU Global
Lin Chieh

C/C++ Project Generator 1.2.4
Generate C/C++ Projects
danielpinto8zz6

C/C++ ms-vscode.cpptools
Microsoft | 17,742,842
C/C++ IntelliSense, debugging, a...

C/C++ for Visual Studio Code

Repository | Issues | Documentation | Code Samples | Offline Installers

Live Share enabled

The C/C++ extension adds language support for C/C++ to Visual Studio Code, including features such as IntelliSense and debugging.

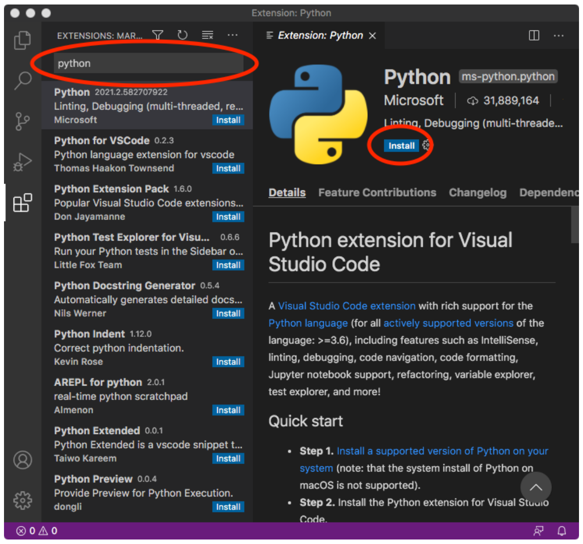
Overview and tutorials

- C/C++ extension overview

C/C++ extension tutorials per compiler and platform

- Microsoft C++ compiler (MSVC) on Windows

(a) C Extension



Python 2021.2.582707922
Linting, Debugging (multi-threaded, re...
Microsoft

Python for VSCode 0.2.3
Python language extension for vscode
Thomas Haakon Townsend

Python Extension Pack 1.6.0
Popular Visual Studio Code extensions...
Don Jayamanne

Python Test Explorer for Visu... 0.6.6
Run your Python tests in the Sidebar o...
Little Fox Team

Python Docstring Generator 0.5.4
Automatically generates detailed docs...
Nils Werner

Python Indent 1.12.0
Correct python indentation.
Kevin Rose

AREPL for python 2.0.1
real-time python scratchpad
Almenon

Python Extended 0.0.1
Python Extended is a vscode snippet t...
Taiwo Kareem

Python Preview 0.0.4
Provide Preview for Python Execution.
dongli

Python ms-python.python
Microsoft | 31,889,164
Linting, Debugging (multi-threade...

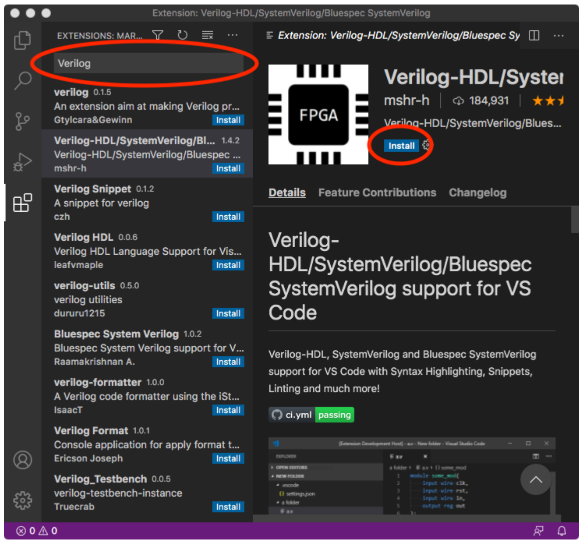
Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.6), including features such as IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, and more!

Quick start

- **Step 1.** Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported).
- **Step 2.** Install the Python extension for Visual Studio Code.

(b) Python Extension



verilog 0.1.5
An extension aim at making Verilog pr...
Gtylcara&Gewinn

Verilog-HDL/SystemVerilog/Bl... 1.4.2
Verilog-HDL/SystemVerilog/Bluespec ...
mshr-h

Verilog Snippet 0.1.2
A snippet for verilog
czh

Verilog HDL 0.0.6
Verilog HDL Language Support for Vis...
leafvmaple

verilog-utils 0.5.0
verilog utilities
dururu1215

Bluespec System Verilog 1.0.2
Bluespec System Verilog support for V...
Raamakrishnan A.

verilog-formatter 1.0.0
A Verilog code formatter using the iSt...
IsaacT

Verilog Format 1.0.1
Console application for apply format t...
Ericson Joseph

Verilog_Testbench 0.0.5
verilog-testbench-instance
Truecrab

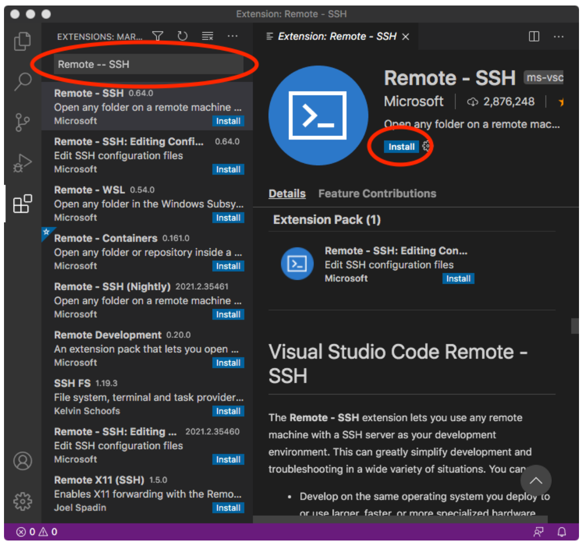
Verilog-HDL/Syste
mshr-h | 184,931 ★★★
Verilog-HDL/SystemVerilog/Blues...

Verilog-HDL/SystemVerilog/Bluespec SystemVerilog support for VS Code

Verilog-HDL, SystemVerilog and Bluespec SystemVerilog support for VS Code with Syntax Highlighting, Snippets, Linting and much more!

ci.yml passing

(c) Verilog Extension



Remote - SSH 0.64.0
Open any folder on a remote machine ...
Microsoft

Remote - SSH: Editing Confi... 0.64.0
Edit SSH configuration files
Microsoft

Remote - WSL 0.54.0
Open any folder in the Windows Subsy...
Microsoft

Remote - Containers 0.161.0
Open any folder or repository inside a ...
Microsoft

Remote - SSH (Nightly) 2021.2.35461
Open any folder on a remote machine ...
Microsoft

Remote Development 0.20.0
An extension pack that lets you open ...
Microsoft

SSH FS 1.19.3
File system, terminal and task provider...
Kelvin Schoofs

Remote - SSH: Editing ... 2021.2.35460
Edit SSH configuration files
Microsoft

Remote X11 (SSH) 1.5.0
Enables X11 forwarding with the Remo...
Joel Spadin

Remote - SSH ms-vsc
Microsoft | 2,876,248
Open any folder on a remote mac...

Extension Pack (1)

Remote - SSH: Editing Con...
Edit SSH configuration files
Microsoft

Visual Studio Code Remote - SSH

The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:

- Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware

(d) Remote - SSH Extension

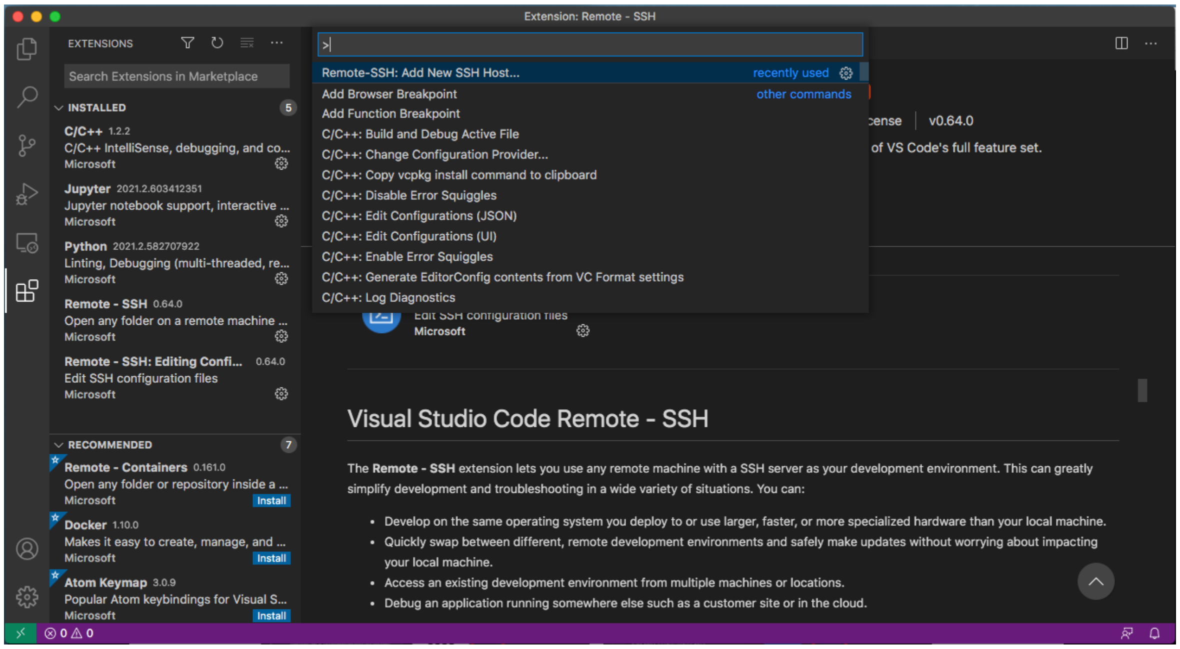
Figure 4: Installing Four Extensions in VS Code

```
C:\Users\username\.ssh\config
```

Or something like this on Mac OS X:

```
/Users/username/.ssh/config
```

A little pop up dialog box in the lower right-hand corner might say *Host added!*. You should now be all set to log into the `ecelinux` servers.


Extension: Remote - SSH
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 5
C/C++ 1.2.2
C/C++ IntelliSense, debugging, and co...
Microsoft
Jupyter 2021.2.603412351
Jupyter notebook support, interactive ...
Microsoft
Python 2021.2.582707922
Linting, Debugging (multi-threaded, re...
Microsoft
Remote - SSH 0.64.0
Open any folder on a remote machine ...
Microsoft
Remote - SSH: Editing Confi... 0.64.0
Edit SSH configuration files
Microsoft
RECOMMENDED 7
Remote - Containers 0.161.0
Open any folder or repository inside a ...
Microsoft Install
Docker 1.10.0
Makes it easy to create, manage, and ...
Microsoft Install
Atom Keymap 3.0.9
Popular Atom keybindings for Visual S...
Microsoft Install
>|
Remote-SSH: Add New SSH Host... recently used
Add Browser Breakpoint other commands
Add Function Breakpoint
C/C++: Build and Debug Active File
C/C++: Change Configuration Provider...
C/C++: Copy vcpkg install command to clipboard
C/C++: Disable Error Squiggles
C/C++: Edit Configurations (JSON)
C/C++: Edit Configurations (UI)
C/C++: Enable Error Squiggles
C/C++: Generate EditorConfig contents from VC Format settings
C/C++: Log Diagnostics
cense v0.64.0
of VS Code's full feature set.
Edit SSH configuration files
Microsoft
⊗ 0 ⚠ 0

Visual Studio Code Remote - SSH

The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:

- Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware than your local machine.
- Quickly swap between different, remote development environments and safely make updates without worrying about impacting your local machine.
- Access an existing development environment from multiple machines or locations.
- Debug an application running somewhere else such as a customer site or in the cloud.

Figure 5: Logging into `ecelinux` Servers with VS Code

6.3. Logging into `ecelinux` Servers with VS Code

After starting VS Code, choose *View > Command Palette* from the menubar. Enter the following command in the command palette:

```
Remote-SSH: Connect Current Window to Host..
```

As you start typing matching commands will be displayed and you can just click the command when you see it. VS Code will then ask you to *Select configured SSH host or enter user@host*, and you should enter the following:

```
ecelinux.ece.cornell.edu
```

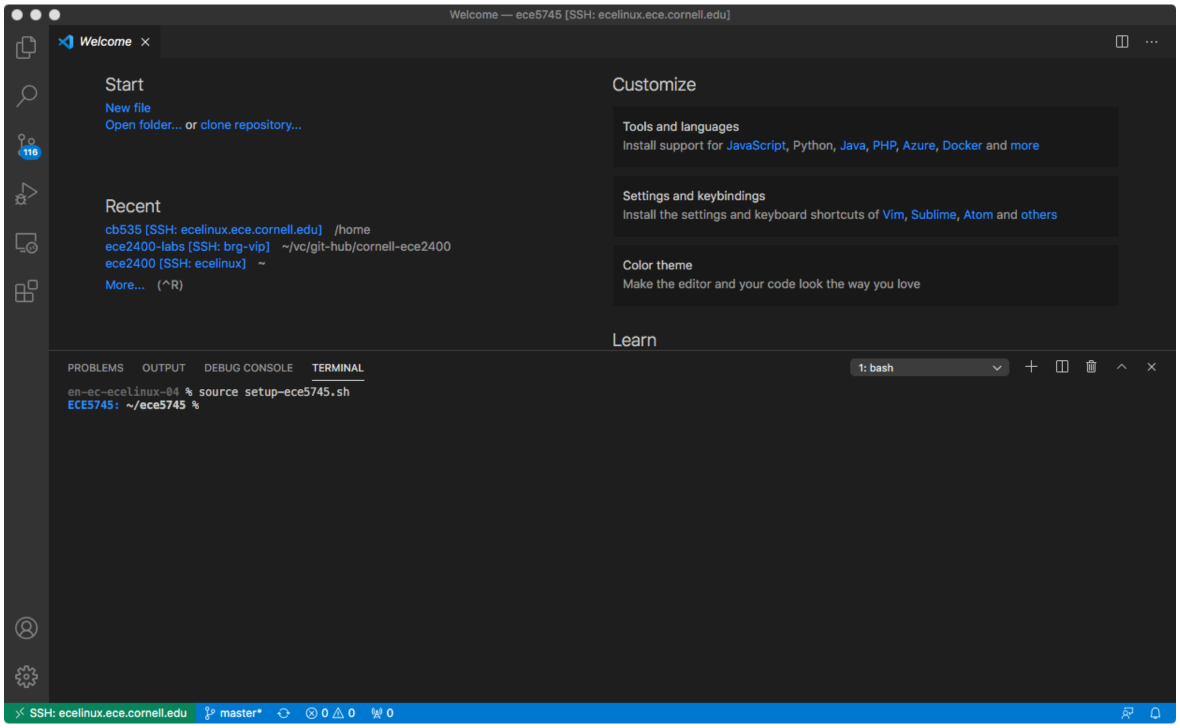
VS Code will prompt you to enter your Cornell NetID password, and then you should be connected to the `ecelinux` servers.

The very first time you log into the `ecelinux` servers you may see a warning like this:

```
"ecelinux.ece.cornell.edu" has fingerprint
"SHA256:smwMnf9dyhs5zW5I279C5oJBrTFc5FLghIJMfBR1cxI".
Are you sure you want to continue?
Continue
Cancel
```

The very first time you log into the `ecelinux` servers it is okay to click *Continue*, but from then on if you continue to receive this warning please contact the course staff.

Also the very first time you log into the `ecelinux` servers you will see a pop up dialog box in the lower right-hand corner which says *Setting up SSH host ecelinux.ece.cornell.edu (details) Initializing....* It


Welcome — ece5745 [SSH: ecelinux.ece.cornell.edu]
Welcome
Start
New file
Open folder... or clone repository...
Recent
cb535 [SSH: ecelinux.ece.cornell.edu] /home
ece2400-labs [SSH: brg-vip] ~/vc/git-hub/cornell-ece2400
ece2400 [SSH: ecelinux] ~
More... (^R)
Customize
Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love
Learn
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
SSH: ecelinux.ece.cornell.edu
master*
0 0
0

```
en-ec-ecelinux-04 % source setup-ece5745.sh
ECE5745: ~/ece5745 %
```

Figure 6: VS Code After Logging into `ecelinux`

might take up to a minute for everything to be setup; please be patient! Once the pop up dialog box goes away and you see *SSH: ecelinux.ece.cornell.edu* in green in the lower left-hand corner of VS Code then you know you are connected to the `ecelinux` servers.

The next step is to open a terminal which will give you access to the Linux command line on the `ecelinux` servers. Choose *Terminal > New Terminal* from the menubar. You should see the same kind of Linux command line prompt that you saw when using either PowerShell or Mac Terminal. The very first thing you need to do after logging into the `ecelinux` servers is source the course setup script. This will ensure your environment is setup with everything you need for working on the laboratory assignments. Enter the following command on the command line:

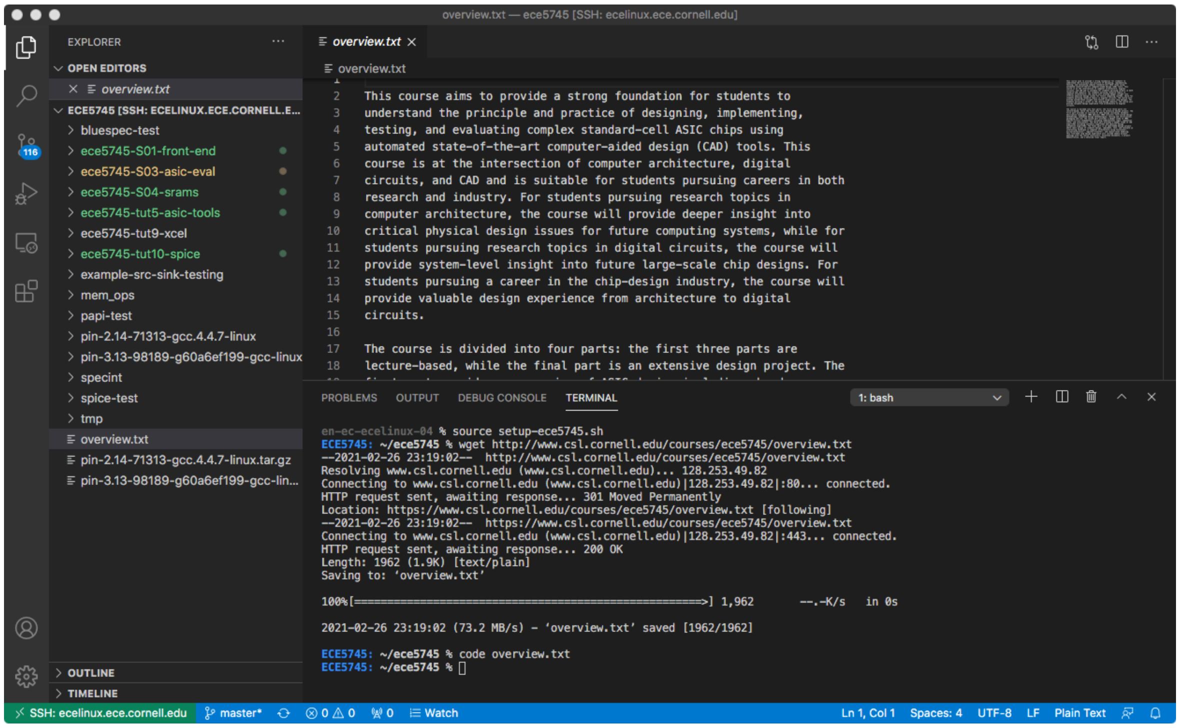
```
 % source setup-ece4750.sh
```

Again, you should not enter the `%` character. You should now see a blue `ECE 4750` in your prompt which means your environment is setup for the course. Figure 6 shows what your prompt should look like if you have sourced the course setup script. See Section 11 for more on how to automatically source the setup script every time you log into the `ecelinux` servers.

6.4. Using VS Code

VS Code has an integrated text editor which makes it very productive for remote development. Log into the `ecelinux` servers and open a terminal as described in the previous subsection. To experiment with VS Code, we will first grab a text file using the `wget` command. The next tutorial discusses this command in more detail.

```
 % wget http://www.csl.cornell.edu/courses/ece4750/overview.txt
```



```

This course aims to provide a strong foundation for students to
understand the principle and practice of designing, implementing,
testing, and evaluating complex standard-cell ASIC chips using
automated state-of-the-art computer-aided design (CAD) tools. This
course is at the intersection of computer architecture, digital
circuits, and CAD and is suitable for students pursuing careers in both
research and industry. For students pursuing research topics in
computer architecture, the course will provide deeper insight into
critical physical design issues for future computing systems, while for
students pursuing research topics in digital circuits, the course will
provide system-level insight into future large-scale chip designs. For
students pursuing a career in the chip-design industry, the course will
provide valuable design experience from architecture to digital
circuits.

The course is divided into four parts: the first three parts are
lecture-based, while the final part is an extensive design project. The
```

```
en-ec-ecelinux-04 % source setup-ece5745.sh
ECE5745: ~/ece5745 % wget http://www.csl.cornell.edu/courses/ece5745/overview.txt
--2021-02-26 23:19:02--  http://www.csl.cornell.edu/courses/ece5745/overview.txt
Resolving www.csl.cornell.edu (www.csl.cornell.edu)... 128.253.49.82
Connecting to www.csl.cornell.edu (www.csl.cornell.edu)|128.253.49.82|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.csl.cornell.edu/courses/ece5745/overview.txt [following]
--2021-02-26 23:19:02--  https://www.csl.cornell.edu/courses/ece5745/overview.txt
Connecting to www.csl.cornell.edu (www.csl.cornell.edu)|128.253.49.82|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1962 (1.9K) [text/plain]
Saving to: ‘overview.txt’

100%[=====================================================>] 1,962       --.-K/s   in 0s

2021-02-26 23:19:02 (73.2 MB/s) - ‘overview.txt’ saved [1962/1962]

ECE5745: ~/ece5745 % code overview.txt
ECE5745: ~/ece5745 % 
```

Figure 7: Typical VS Code Session with Terminal, Open File, and File Explorer

You can open a file in the integrated text editor using the `code` command like this:

```
 % code overview.txt
```

Notice how the `overview.txt` file opened in a new tab at the top and the terminal remains at the bottom. This enables you to have easy access to editing files and the Linux command line at the same time.

VS Code also has an integrated file explorer which makes it very productive to browse and open files. Choose *View* > *Explorer* from the menubar, and then click on *Open Folder*. VS Code will then ask you to *Open File Or Folder*, and you should enter the following:

```
  /home/netid
```

Replace `netid` with your Cornell NetID in the path above. Click *OK*. This will reload VS Code so the terminal and open files will go away, but now you will see a file explore in the left sidebar. You can easily browse your directory hierarchy, open files by clicking on them, create new files, and delete files. Go ahead and reopen the terminal using *Terminal* > *New Terminal* and open the `overview.txt` file again using the `code` command. Figure 7 is a screenshot of what your VS Code session should look like.

The final step is to make sure your extensions for C/C++, Python, and Verilog are also installed on the server. Choose *View* > *Command Palette* from the menubar. Search for the same C/C++, Python, and Verilog extensions we installed earlier. When you find these extensions instead of saying *Install* it should now say *Install in SSH: ecelinux.ece.cornell.edu*. Install all three language extensions on the `ecelinux` servers.

6.5. Troubleshooting Remote Access via VS Code

There may be issues where sometimes VS Code just keeps asking you for your password or VS Code just hangs when you try and connect to the `ecelinux` servers. This might mean VS Code is getting wedged. You can definitely ask the course staff to help, but you can also try to fix it on your own. One thing to try is to delete the `.vscode-server` directory on the sever. Of course, how can you delete this directory if you cannot use VS Code to access the `ecelinux` servers? You can use PowerShell (see Section 4) or Mac Terminal (see Section 5) to log into the `ecelinux` servers.

Once you have gained access to the Linux command line on the `ecelinux` servers using either PowerShell or Mac Terminal, then you can delete the `.vscode-server` directory like this:

```
 % rm -rf .vscode-server
```

7. ~~Remote Access via X2Go~~

The X2Go remote access option is no longer supported on the `ecelinux` servers. Keeping this comment here so you don't think we're gaslighting you. (Will remove for Fall 2025.)

8. Remote Access via MobaXterm

Note: we believe you can accomplish everything you need to with VS Code. You are welcome to try out MobaXterm, but the staff may not be able to support you if you run into technical difficulties. (Basically every TA in this class used VS Code exclusively and we think you should, too.)

Section 4 described how to access the `ecelinux` servers using PowerShell, but this option does not permit using a Linux application with a GUI. If you have not tried the remote access option described in Section 4 then do that now. Linux uses the X11 client/server system to manage GUI applications, so to use a Linux application with a GUI we need to setup an X11 client/server. On Windows we will install the MobaXterm client/server system. Once MobaXterm is installed then we can use it to log into the `ecelinux` servers and start GUI applications. **Note that we actually recommend using X2Go as described in Section 7 if you want to use a Linux application with a GUI, but some students may prefer MobaXterm.**

8.1. Installing MobaXterm on Your Windows Laptop/Workstation

Go to this page, download the installer, then install MobaXterm.

- `http://mobaxterm.mobatek.net/download-home-edition.html`

We recommend using the *Installer Edition*. Right click on the downloaded ZIP archive and choose *Extract all*. Then double click on `MobaXterm_installer`, click *Next* to start the installation, agree to the EULA and click *Next*, click *Next* to install in the default location, click *Install* to start the installation, and click *Finish* to finish the installation.

8.2. Starting and Configuring MobaXterm

First, if you are off campus, then you must be connected to the Cornell VPN before attempting to use MobaXterm to access the `ecelinux` servers (see Section 2). To start MobaXterm click the *Start* menu then choose *All Programs > MobaXterm > MobaXterm*, or click the *Start* menu, type *MobaXterm*, and choose *MobaXterm*.

We recommend setting the SSH "keep alive signal" to avoid dropping your connection if you are inactive for a certain amount of time. After starting MobaXterm, choose Settings > Configuration from the menu, select the *SSH* tab, and make sure the *SSH keepalive* option is checked.

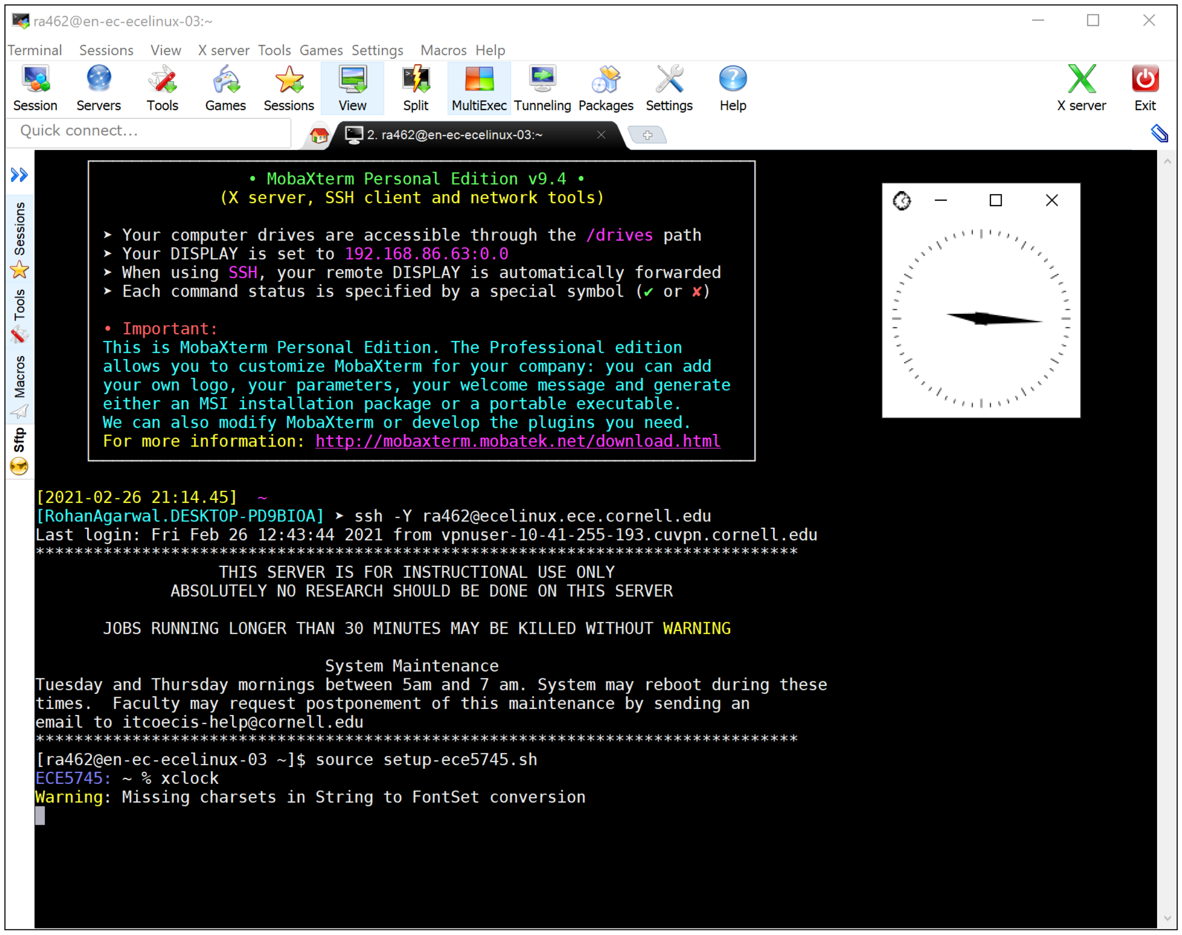
8.3. Logging into `ecelinux` Servers with MobaXterm

After starting MobaXterm, type the following to configure your local laptop/workstation to avoid dropping your connection if you are inactive for a certain amount of time.

```
% echo "Host *.ece.cornell.edu"    >> ~/.ssh/config
% echo "  ServerAliveInterval 180" >> ~/.ssh/config
```

Then type in the following command at the prompt to log into the `ecelinux` servers using SSH.

```
% ssh -Y netid@ecelinux.ece.cornell.edu
```


ra462@en-ec-ecelinux-03:~
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. ra462@en-ec-ecelinux-03:~
Sessions Tools Macros Sftp

```
• MobaXterm Personal Edition v9.4 •
        (X server, SSH client and network tools)

➤ Your computer drives are accessible through the /drives path
➤ Your DISPLAY is set to 192.168.86.63:0.0
➤ When using SSH, your remote DISPLAY is automatically forwarded
➤ Each command status is specified by a special symbol (✔ or ✘)

• Important:
This is MobaXterm Personal Edition. The Professional edition
allows you to customize MobaXterm for your company: you can add
your own logo, your parameters, your welcome message and generate
either an MSI installation package or a portable executable.
We can also modify MobaXterm or develop the plugins you need.
For more information: http://mobaxterm.mobatek.net/download.html

[2021-02-26 21:14.45]  ~
[RohanAgarwal.DESKTOP-PD9BIOA] ➤ ssh -Y ra462@ecelinux.ece.cornell.edu
Last login: Fri Feb 26 12:43:44 2021 from vpnuser-10-41-255-193.cuvpn.cornell.edu
*******************************************************************************
                   THIS SERVER IS FOR INSTRUCTIONAL USE ONLY
              ABSOLUTELY NO RESEARCH SHOULD BE DONE ON THIS SERVER

       JOBS RUNNING LONGER THAN 30 MINUTES MAY BE KILLED WITHOUT WARNING

                              System Maintenance
Tuesday and Thursday mornings between 5am and 7 am. System may reboot during these
times.  Faculty may request postponement of this maintenance by sending an
email to itcoecis-help@cornell.edu
*******************************************************************************
[ra462@en-ec-ecelinux-03 ~]$ source setup-ece5745.sh
ECE5745: ~ % xclock
Warning: Missing charsets in String to FontSet conversion
```

Figure 8: MobaXterm After Logging Into `ecelinux` Servers and Starting `xclock`

Replace `netid` with your Cornell NetID in the command above. This is very similar to what was discussed in Section 4 with the key difference being the `-Y` command line option which enables the X11 client/server system. This system means when you start a GUI application on the `ecelinux` server, the GUI application is displayed on your laptop/workstation, but the GUI application itself is running on the `ecelinux` servers.

As always, once you have opened a terminal, the very first thing you need to do after logging into the `ecelinux` servers is source the course setup script.

```
% source setup-ece4750.sh
```

You should now see a blue `ECE 4750` in your prompt which means your environment is setup for the course. See Section 11 for more on how to automatically source the setup script every time you log into the `ecelinux` servers, and see Section 10 for more on text editors you can use in the course.

To test out the X11 client/server system, we will start a very basic GUI analog clock application on the `ecelinux` server. Type in the following command at the prompt to start the clock application.

```
% xclock
```

Hopefully, you should see a window pop up on your local laptop/workstation displaying an analog clock window. It is critical to understand that while the GUI application is being *displayed* on your local laptop/workstation, the GUI application itself is running on the `ecelinux` servers. Figure 8 shows what your prompt should look like if you have sourced the course setup script and what the analog clock window looks like.

After logging into an `ecelinux` server you will notice that a sidebar appears on the left that shows you the files in your home directory. You can drag files to/from the sidebar and the desktop to easily move files to/from your local Windows laptop/workstation and the `ecelinux` servers. To hide the sidebar choose View > Show/hide sidebar from the menu.

9. Remote Access via Mac Terminal with XQuartz

Note: we believe you can accomplish everything you need to with VS Code. You are welcome to try out XQuartz, but the staff may not be able to support you if you run into technical difficulties. (Basically every TA in this class used VS Code exclusively and we think you should, too.)

Section 5 described how to access the `ecelinux` servers using Mac Terminal, but this option does not permit using a Linux application with a GUI. If you have not tried the remote access option described in Section 5 then do that now. Linux uses the X11 client/server system to manage GUI applications, so to use a Linux application with a GUI we need to setup an X11 client/server. On Mac OS X we will install the XQuartz client/server system. Once XQuartz is installed then we can use Mac Terminal with a slightly different command line to log into the `ecelinux` servers. **Note that we actually recommend using X2Go as described in Section 7 if you want to use a Linux application with a GUI, but some students may prefer Mac Terminal with XQuartz.**

9.1. Installing XQuartz on Your Mac OS X Laptop/Workstation

Go to this page, download the DMG, and then install XQuartz.

- `https://www.xquartz.org`

After installing, you should reboot your laptop/workstation and then try starting XQuartz. The very first time you might need to right click on the XQuartz application and explicitly choose *open* from the pop-up menu. This tells Mac OS X that you really do want to run this application even though you downloaded it from the internet. After doing this once, you can open XQuartz just by double clicking it, and/or XQuartz will start automatically when necessary.

9.2. Starting and Configuring Mac Terminal

First, if you are off campus, then you must be connected to the Cornell VPN before attempting to use X2Go to access the `ecelinux` servers (see Section 2). To start Mac Terminal go to your *Applications* folder and choose *Utilities* > *Terminal.app*, or open Spotlight, type *Terminal*, and press enter.

We recommend setting the SSH "keep alive signal" to avoid dropping your connection if you are inactive for a certain amount of time. After starting Mac Terminal, type the following to configure your local laptop/workstation to avoid dropping your connection if you are inactive for a certain amount of time.

```
% echo "Host *.ece.cornell.edu"    >> ~/.ssh/config
% echo "  ServerAliveInterval 180" >> ~/.ssh/config
```

You should only need to do this once.

9.3. Logging into `ecelinux` Servers with Mac Terminal

After starting Mac Terminal, type in the following command at the prompt to log into the `ecelinux` servers using SSH.

```
 % ssh -Y netid@ecelinux.ece.cornell.edu
```

Replace `netid` with your Cornell NetID in the command above. This is very similar to what was discussed in Section 5 with the key difference being the `-Y` command line option which enables the X11 client/server system. This system means when you start a GUI application on the `ecelinux` server, the GUI application is displayed on your laptop/workstation, but the GUI application itself is running on the `ecelinux` servers.

As always, once you have opened a terminal, the very first thing you need to do after logging into the `ecelinux` servers is source the course setup script.

```
 % source setup-ece4750.sh
```

You should now see a blue `ECE 4750` in your prompt which means your environment is setup for the course. See Section 11 for more on how to automatically source the setup script every time you log into the `ecelinux` servers, and see Section 10 for more on text editors you can use in the course.

To test out the X11 client/server system, we will start a very basic GUI analog clock application on the `ecelinux` server. Type in the following command at the prompt to start the clock application.

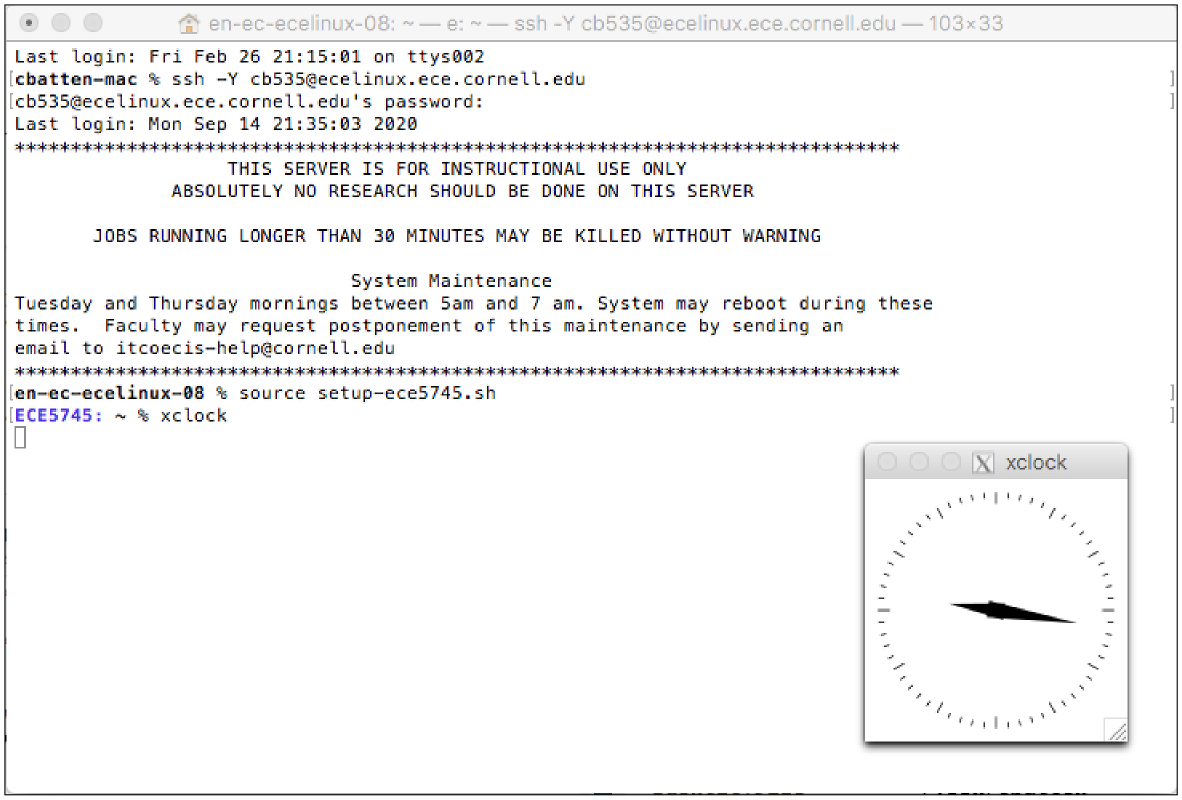
```
 % xclock
```

Hopefully, you should see a window pop up on your local laptop/workstation displaying an analog clock window. It is critical to understand that while the GUI application is being *displayed* on your local laptop/workstation, the GUI application itself is running on the `ecelinux` servers. Figure 9 shows what your prompt should look like if you have sourced the course setup script and what the analog clock window looks like.

10. Linux Text Editors

Once you have successfully logged into the `ecelinux` servers, gained access to the Linux command line, and sourced the setup script, one of the first things you will want to do is choose a text editor for editing code. If you have not gone through one of the remote access options described in Sections 4–9 then do that now. There are two kinds of editors: text-based editors and GUI-based editors. The *text-based editors* work by opening files through the command line and then using the keyboard to navigate the file system, open files, save files, search within files, etc. The *GUI-based editors* work by providing a graphical user interface so that the user can use a mouse to interact with the editor. If you have chosen to use the PowerShell (see Section 4) or Mac Terminal (see Section 5) remote access options then you must use a text-based editor (e.g., nano). If instead you have used one of the other remote access options then you can use either a text-based editor or a GUI-based editor. Note that if you have chosen to use the VS Code remote access option then there is no need to use a Linux text editor; you can directly use VS Code to remotely edit files on the `ecelinux` servers as described in Section 6.

To experiment with these editors, we will first grab a text file using the `wget` command. The next tutorial discusses this command in more detail.


en-ec-ecelinux-08: ~ — e: ~ — ssh -Y cb535@ecelinux.ece.cornell.edu — 103×33
xclock

```
Last login: Fri Feb 26 21:15:01 on ttys002
cbatten-mac % ssh -Y cb535@ecelinux.ece.cornell.edu
cb535@ecelinux.ece.cornell.edu's password:
Last login: Mon Sep 14 21:35:03 2020
*******************************************************************************
                   THIS SERVER IS FOR INSTRUCTIONAL USE ONLY
              ABSOLUTELY NO RESEARCH SHOULD BE DONE ON THIS SERVER

       JOBS RUNNING LONGER THAN 30 MINUTES MAY BE KILLED WITHOUT WARNING

                              System Maintenance
Tuesday and Thursday mornings between 5am and 7 am. System may reboot during these
times.  Faculty may request postponement of this maintenance by sending an
email to itcoecis-help@cornell.edu
*******************************************************************************
en-ec-ecelinux-08 % source setup-ece5745.sh
ECE5745: ~ % xclock
```

Figure 9: Mac Terminal with XQuartz After Logging Into `ecelinux` Servers and Starting `xclock`

```
% wget http://www.csl.cornell.edu/courses/ece4750/overview.txt
```

10.1. Using the Nano Text Editor

Nano is a very simple non-graphical text editor installed on the `ecelinux` machines. The editor is easy to learn and use. You can start Nano by typing the command `nano` in the terminal and optionally specifying the filename you want to view and edit.

```
% nano overview.txt
```

Use the arrow keys to move the cursor position. Notice that the editor specifies most of the useful commands at the bottom of the terminal screen. The symbol ^ indicates the `CONTROL` key. To type any text you want, just move the cursor to the required position and use the keyboard. To save your changes press `CONTROL+O` and press the `<ENTER>` key after specifying the filename you want to save to. You can exit by pressing `CONTROL+X`.

10.2. Other Editors

While nano is easy to learn, students that anticipate using Linux in the future beyond this course might want to use a more powerful text-based editor such as `emacs` or `vim`. Both also have GUI equivalents. By default `emacs` will start the GUI, while using the `-nw` command line option will enable non-graphical mode. `vim` is purely non-graphical. It is beyond the scope of this tutorial to teach you the usage of these editors, but most advanced Linux users use one of these more powerful text editors for development.

11. Sourcing the Course Setup Script with Auto Setup

Each of the previous sections have demonstrated how to remotely access the `ecelinux` servers, get to the Linux command line, and source the course setup script. **Again, you must source the course setup script before doing any work related to this course!** The course setup script configures everything so you have the right environment to work on the laboratory assignments.

Since it can be tedious to always remember to source the course setup script, you can also use auto setup which will automatically source the course setup for you when you open a terminal. Note that if the environment for ECE 4750 conflicts with the environment required by a different course then you will need to manually source the setup script when you are working on this course. Enter the following command on the command line to use auto setup:

```
 % source setup-ece4750.sh --enable-auto-setup
```

Now completely logout of the `ecelinux` servers then log back in and get to the Linux command line. The exact way to do this depends on which remote access option you using. You should see `ECE 4750` in the prompt meaning your environment is automatically setup for the course. If at anytime you need to disable auto setup you can use the following command:

```
 % source setup-ece4750.sh --disable-auto-setup
```

Again, if for any reason running the setup script prevents you from using tools for another course, you cannot use the auto setup. You will need to run the setup script manually every time you want to work on a laboratory assignment for this course.

Note: some users have had trouble using these flags to enable or disable the auto-setup. If the `-enable-auto-setup` or `-disable-auto-setup` flags don't seem to be working, you can enable by manually adding the source line to your `.bashrc` file and you can disable by manually removing the source line from your `.bashrc`. Please reach out to a staff member and they will help you with this!

Acknowledgments

This tutorial was developed for the ECE 2400 Computer Systems Programming and ECE 5745 Complex Digital ASIC Design courses at Cornell University by Christopher Batten.