Setting up a computing environment for C like mine.

There are a lot of things you can do to set up you computer for this course. The document covers what I did. Feel free to copy me. If you do not, you are largely on your own. (If you do copy me, you are only slightly less on your own.) It is certainly possible to set up a c compiler on your local machine an an alternative to following these directions. It is also possible to do everything on the CS unix servers and use your own machine simply as a terminal. I have taken all of these approaches (and others) at times during the past few years. This is what I currently do. Be aware that all programs MUST compile and run on the Unix servers. So even if you do everything locally, before turning in assignments you will probably want to test on Unix.

Please note: this is a really long configuration process, with many places to go wrong. I have checked this on both windows and Macs, but it is possible that I missed some steps. I encourage you to follow these directions slowly. Do you best to understand, not just type. (I know that there is a lot here that you will not understand. Every step should make sense by the end of the semester.)

Watch out for "." within the shell commands below. "." is a significant character and should not be missed.

TYPOGRAPHY: I attempt below to always put commands to be typed into a terminal on lines by themselves and in the `menlo font.`

Step 1:
Create a Public/Private key
(These steps were updated 6pm on 2/12) to insert a new step at step 9.

1. Open a terminal window (You will use terminals a lot this semester.)
    1. On a mac: Start the Terminal program in Applications/Utilities
    2. On Windows: Open a windows powershell (type powershell in the search box in lower left then enter)
2. Enter the following command to start the key generation process
   ```
   ssh-keygen -t rsa
   ```
3. Press the ENTER key to accept the default location.
4. The ssh-keygen utility will then prompt you for a passphrase. **Press enter to give no passphrase**. The point of creating the key is so that you do not need to enter a pass phrase all the time
5. cd to the location you were told was the default location. Usually this will work
   ```
   cd .ssh
   ```
5. In this directory you should see id_rsa and id_rsa.pub. These are your private and public keys
6. Execute the following command
   ```
   scp id_rsa.pub YOUR_NAME@165.106.10.186:mpub
   ```
7. ssh into that machine
   ```
   ssh YOUR_NAME@165.106.10.186
   ```
8. on …186
   ```
   mkdir .ssh
   cd .ssh
   ```
9. if this directory contains the file authorized_keys then on …186
   ```
   cat ~/mpub >> authorized_keys
   ```

else
```
cp ~/mpub authorized_keys
```
10. exit …186, on …186
```
exit
```
11. log back into …186 as in step 7. If everything worked correctly you should not have to enter a password. This is what you want! If you are still asked for a password, you did something wrong.

Step 2 (optional):
(This step works on Mac/Unix, it might not work on Windows)
Add some ssh configuration
Within the .ssh directory on your local machine create a file named config
Into that file put the following:

```
Host lab186
    HostName 165.106.10.186
    ForwardX11 yes
    ForwardX11Trusted yes
    User YOUR_UNIX_NAME
```

test this configuration. In a terminal/powershell
```
ssh lab186
```

The configuration saves you a little extra typing. (There is a lot more you can do with configuration files. Feel free to explore. Also take a look at my config file /home/gtowell/Public/246/config. There are actually 10 lab machines available. My config sets up a shortcut to each.)

Step 3:
Get and install Visual Studio Code (VSC)
https://code.visualstudio.com/download

Step 4.

Add extensions to VSC
The only extensions I use for 246 are:
C/C++ from Microsoft (v1.1.3) Other version numbers are likely OK
SFTP by lixmomo v1.12.9
Extensions are added/changed by tapping on the icon at left which is on the left edge of the VSC window.

Step 5.
C include files. C include files are similar to Java includes except that are actual separate files. You will learn more about them in class. For now, just know that you will need them and you do not have them on your personal machine. You do not really need them on your machines, but VSC will be a lot happier with them there.

open powershell/terminal
```
cd ~
scp main186:/home/gtowell/Public/246/includee.tar .
tar fvx includee.tar
rm includee.tar
```

Step 6.

Make a directory where all of your work will live

open a terminal/powershell and connect to main186 using ssh (you can use the terminal from step 5 if it is still open)

Create a folder into which all of your 246 work will be put. For example:

```
mkdir 246
```

Leave this terminal open on your device, you will come back to it

Step 7.

Open VSC

Set up a directory in VSC that you will use as the root directory of your coding. You might name this 246 to echo the name on unix

Hit the F1 button

in the search window that appears enter “SFTP:config” then hit return

this will open an editor on a file sftp.json that initially contains just {}

replace the {} with

```
{
   "name": “main186”,
   "host": “165.106.10.186”,
   "protocol": "sftp",
   "port": 22,
   "username": “YOUR UNIX NAME",
   "privateKeyPath": “PATH TO YOUR id_rsa FILE  ON YOUR DEVICE”,
   "remotePath": "246",
   "uploadOnSave": true
}
```

On my mac the private key path is /Users/geoffreytowell/.ssh/id_rsa
On my windows box the private key path is c:\\Users\\towell\\.ssh\\id_rsa
The remote path is the directory you made in step 6 on the lab machines

Your paths will likely be similar.

Step 8:
Configure C/C++ in VSC to use your includes

Within VSC hit F1, then type “c/c++: edit con” then click on the line that ends “(JSON)”

Find

```
"includePath": [
            “${workspaceFolder}/**"
         ],
```

Change this entry to be:

```
 "includePath": [
    "${workspaceFolder}/**",
    "~/include"
 ],
```

Also find

```
”cStandard": “c89",
```

and change it to

```
"cStandard": "c11",
```

This tells VSC to use the include files you installed in step 5 and to use the correct version of C

Step 9. Test your work
Create a new file in your 246 directory in VSC (this will be on your local machine). For instance name the file HW.c and create a hello world program given below.

```
#include <stdio.h>
int main(int argc, char const *argv[])
{
   printf("Hello World\n");
   return 0;
}
```

If you step up the include files correctly you should have no red squiggle underlines. Otherwise you might have several.

Save the program
Go over to the window in which you should still have an open connection to powerpuff

```
cd 246
ls
```

You should see the file HW.c (If it is not there then there is a problem with your SFTP configuration)

Compile and run

```
gcc HW.c
a.out
```

Please note that you can— and indeed should use subfolders. If you create the subfolders within VSC the sftp config will create parallel subfolders on the UNIX side.

Every time you save, the file gets copied over to UNIX. (It you did not create and install the public/private keys correctly, this would get very old, very fast)