Welcome to CS-392 Systems Programming!

Your grade in this class is made up entirely of assignments throughout the semester. You will need a Linux-Lab account on which your assignments must compile and run without any warnings or errors. We will be grading your assignments on the Linux-Lab.

After you have access to the Linux-Lab, you can work on it in three different recommended ways. You may use Putty to log in and write your code using Emacs/Vim. You may work on your own computer and compile your code with your own compiler, then when you are done use FTP or SCP (eg. Filezilla) to transfer your files over to test. You may use an SFTP net drive (Eldos SFTP Net Drive) to have the linux lab mapped to your computer. Whichever way works for you.

Emacs and Vim are text editors you can use inside the terminal. If working locally, other text editors you can check out are Sublime Text, Atom, Visual Studio Code, or Notepad++. All of these are free to use.

For this class you should make a single folder called cs-392 where you put all of your code for the semester. Inside that folder your directory should look like this

```
cs-392
        | include
        | lib
        | src
                | my
                | list
                | etc...
        | test
```

include - This folder should include all your **library** headers.
lib - This is the folder in which you should compile your .a library files into
src - This folder should include all you assignment source code. Inside should be separate folders for each of your assignments (my, list, etc...)
test - You can use this folder to test your library assignments. Your tests will not be graded

For each assignment, please archive your whole cs-392 directory with every assignment in it and submit on canvas. We will rebuild the assignment ourselves.

For all assignments, you are required to compile with the flags -Wall -Werror --pedantic. Optionally, you may also use -std=c99. Code that does not compile will result in a 0.

For your first few assignments you will be building a library. Since you will not learn Makefiles in time you can use these commands to build a library

```
cs-392/src/my > gcc -c *.c -Wall -Werror --pedantic -I../../include
cs-392/src/my > ar -rc libmy.a *.o
cs-392/src/my > ranlib libmy.a
cs-392/src/my > mv libmy.a ../../lib
```

In a library assignment, each function in the library should be in its own file. For example the function void my_char(char) should be inside a file name my_char.c located with in the src/my/ folder. The file should look like this:

```
#include "my.h"

void my_char(char c){
      write(1, &c, 1);
}
```

To test these functions, you will write a program in the test/ folder. For example for you first assignment you might write a 'testmy.c'. This file should have a main.

```
#include "my.h"

int main(int argc, char **argv){
      my_char('h');
      return 0;
}
```

Be aware we do extensive tests on your code. Make sure to include NULL checks and weird inputs. For example:

```
#include "my.h"

int main(int argc, char **argv){
      my_str("");
      my_str(NULL);
      my_str("Hello");
      return 0;
}
```

This should all be acceptable inputs for my_str and should be handled appropriately. That being said, things that C cannot handle (putting a string instead of an Int parameter, using a number or string that is larger than the largest value, or using invalid values) will not be tested.

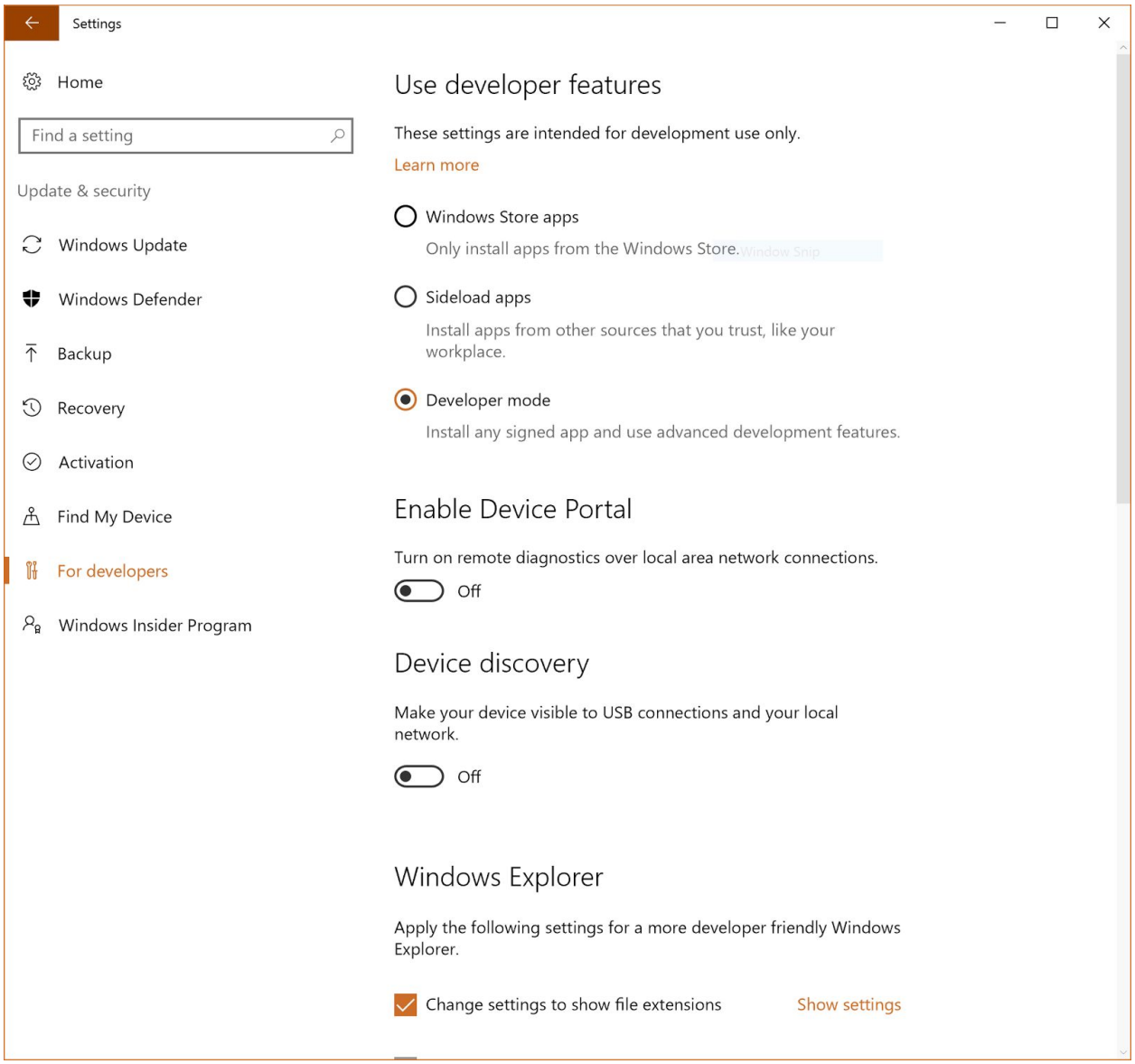
And should be compiled with the command:

```
cs-392/src/my > gcc testmy.c -o testmy -Wall -Werror --pedantic -I../include -L../lib -lmy
```

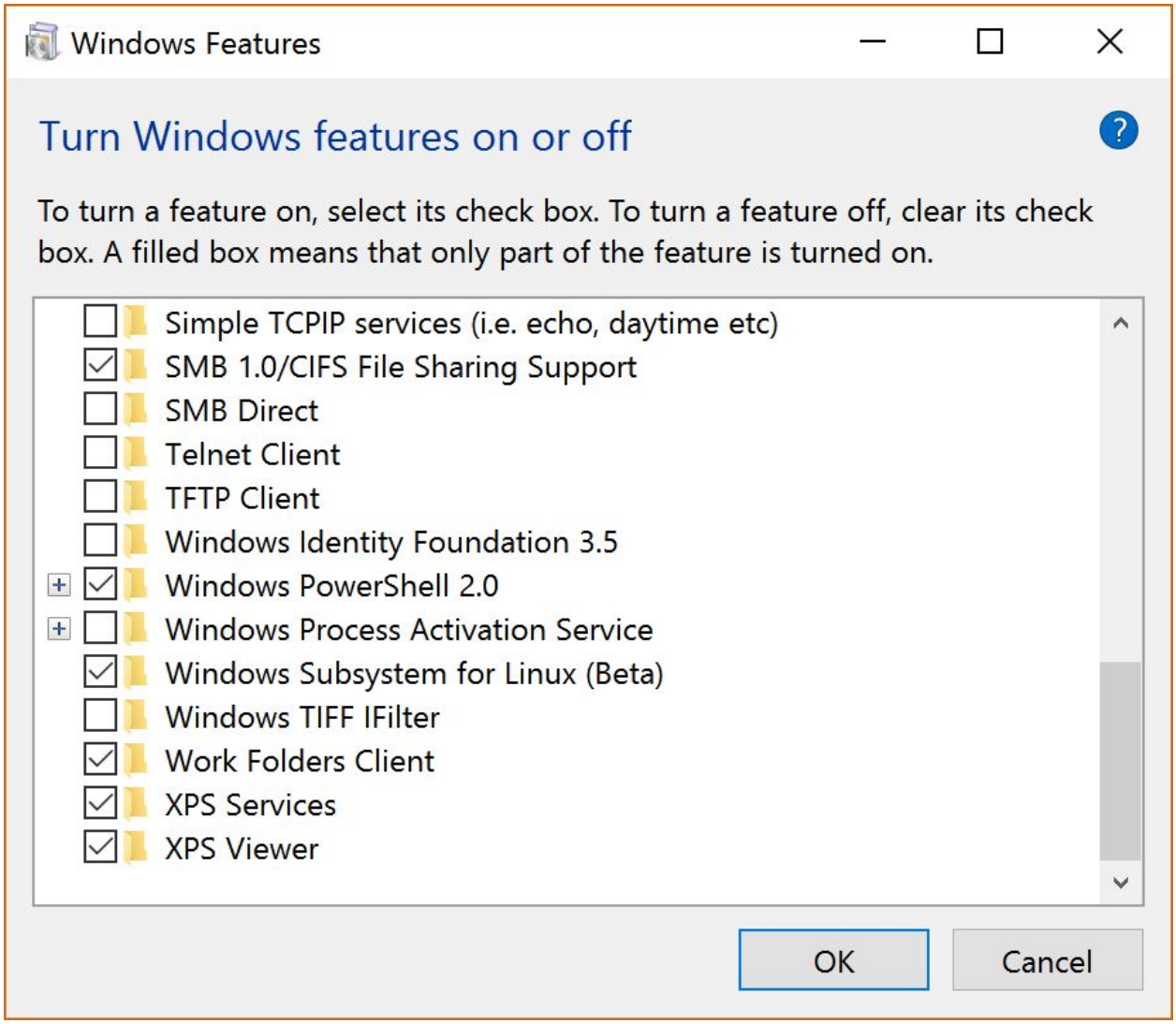
I KNOW this can be a little confusing. Feel free to ask questions if any of this doesn't make sense. It'll get MUCH easier after an assignment or two.

Bash on Ubuntu on Windows

If you have the Windows 10 Anniversary update (v. 1607) then you also have access to an Ubuntu subsystem and a BASH shell. To use this you must first go into **Settings>Update and Security>For Developers** and activate **Developer mode**



Then you use **Turn Windows Feature on or off** and turn on the **Windows Subsystem for Linux**



This will likely require a restart.

Afterwards you will be able to run a BASH command from the Command Prompt and run Linux. You will not however have all of the programs you may need for the class. Use **sudo apt-get gcc** and **sudo apt-get make** to install gcc and make.

A last note. You can access all your Windows files by navigating to /mnt/c/.