Development Environment Setup

You will need four tools to write your programs for this class:

1. **a Compiler** - A program that turns our source code into a program (we will use GCC)
2. **Git** - A version control program to help us track out work and submit work
3. **Make** - A program that makes using the compiler a bit easier
4. **a Text Editor** - What we use to type our code

You can use any text editor you like, but I would recommend Visual Studio Code and that is what I will use throughout the course in demonstrations. You might want to also turn off the telemetry settings in Visual Studio Code as well if you are privacy minded.

macOS Setup

On macOS, Git and GCC come with the Xcode command line tools package. This package must be installed separately from Xcode itself. To see if you already have it installed, open up the Terminal application. When you open up Terminal, you will be presented with a black window with some text that will probably have your username in it and possibly the name of your computer in a format like this before a cursor:

```
Computer-Name:~ Username$
```

The appearance of your terminal could be different, but you should still be able to run commands using this prompt.

When describing commands in the following steps, I will prefix them with a `$` character to indicate that this is a command line instruction. You do not need to type in the first `$`character.

First try running Git by typing the following command and hit enter:

```
$ git
```

If Git is already installed you will see a usage message. If not, a window will pop up asking if you want to install the Xcode command line tools. Install them now if this happens.

Note: If you are running **macOS Big Sur** you might get an error that looks like `xcrun: error:...`. In that case you can run the following command to update your command line tools and fix that issue:

```
$ xcode-select --install
```

After installing the command line tools, you may get the following message:

```
Agreeing to the Xcode/iOS license requires admin privileges, please re-run as root via sudo.
```

If so, run this command:

```
$ sudo xcodebuild -license
```

You will be prompted for a password, enter the password for your computer. You can scroll through the license by pressing the spacebar, and press `q` to stop reading the license. You will then need to type `agree` and press the enter key. Once you have completed this, try running git again.

That is all you need to do to set up your Mac!

Windows Setup

To install your compiler, git, and make, you will need to install Windows Subsystem for Linux (WSL).

These instructions will help you to get WSL setup on your computer. There are two methods you can use to install WSL in the instruction. Make sure to try the **How to install Windows Subsystem for Linux using Settings** method first as it is a bit easier. If the previous method doesn't work, try the instructions using Powershell.

Once you have WSL installed and have rebooted your PC, you can install your Linux distribution from the Windows App Store. There are many distributions to choose from, but make sure to install Ubuntu 20.04 (18.04 should work fine as well).

Once Ubuntu is installed you can open it from your start menu like any other program. If you do not see it on your start menu, you can always hit the Windows key on your keyboard and start typing `Ubuntu` to search for it.

The first time Ubuntu opens it will need to take some time to setup a few additional things. The first thing it will ask you to do is setup your UNIX username and password. The username will need to be lowercase and all one word. **Make sure to remember your login information.** When Ubuntu is finished setting up your account you will be greeted with a simple command line interface (CLI). This should be a line that starts with your username and ends with a `$` and a blinking cursor. From here you can type in commands to your Linux operating system.

When describing commands in the following steps, I will prefix them with a `$` character to indicate that this is a command line instruction. You do not need to type in the first `$`character.

First, lets tell Ubuntu to go and get the latest list of programs and updates from the internet. You may need to use your password to make the following commands work. Also if at any time these command as you are sure you want to do something, you can type `y` and hit the `enter` key. Typing `n` and hitting `enter` will abort the command.

```
$ sudo apt update
```

The command `sudo` tells ubuntu that you would like to run the following command with administrator privileges (that's why you need to type your password). You don't need to use `sudo` for all commands, only for important stuff that affects the operating system like installing programs and updates. The `apt` program is your package manager that helps you install applications, libraries, and other useful things. The `update` option is a part of `apt` and tells that program to go out and update its list of packages.

When that finishes we will tell `apt` to apply any updates it found.

```
$ sudo apt upgrade
```

This could take a while.

Now that we have the latest software for Ubuntu lets install our compiler, make, and git.

```
$ sudo apt install build-essential git
```

When you use the **`install`** option for **`apt`** you can list the names of programs or packages you want to install. The **`build-essential`** package is a metapackage. That is a fancy way for saying that **`build-essential`** is an alias for all the tools that are essential for building programs and installs all of them with one simple phrase (saves a lot of typing). Notice that we also can install **`git`** at the same time by putting a space between the packages and programs we would like to install.

If you decided to use Visual Studio Code as your editor of choice, there is a plugin called **`Remote - WSL`** that might save you some time on your work. It allows your editor to provide you with easier access to Ubuntu in WSL and gives you a convenient embedded terminal to run your code.

Now that you have WSL installed for Windows and the Ubuntu operating system there is another command that may be helpful. With the WSL Ubuntu window open, type the following command:

```
$ explorer.exe .
```

The **`explorer.exe`** command tells Windows to open a Windows File Explorer Window. The **`.`** refers to the current directory in a UNIX path (if you don't know the current directory you can use the command **`pwd`**). This way you can see all your files and folders through a graphical interface instead of just using the command line interface.

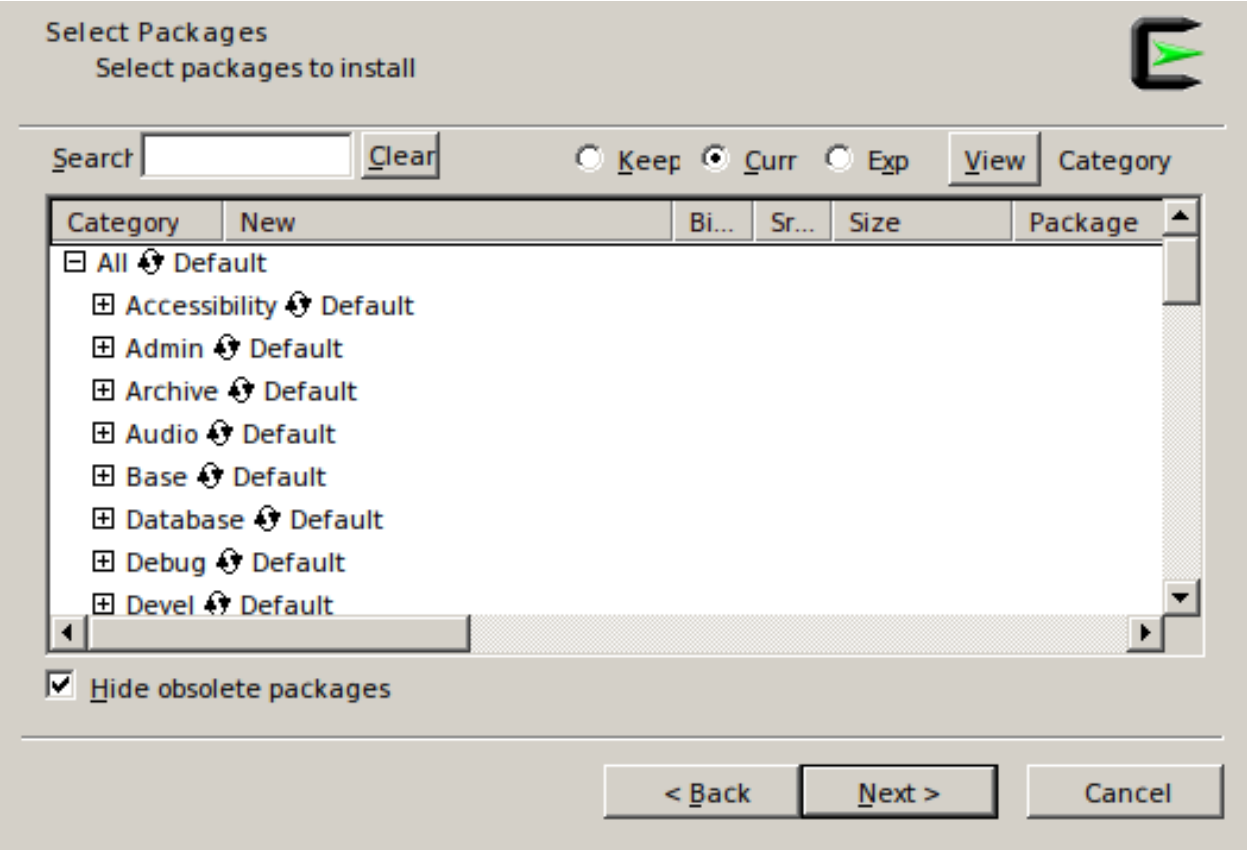
Unable to install WSL?

Only do the following if you are on Windows and you cannot install WSL

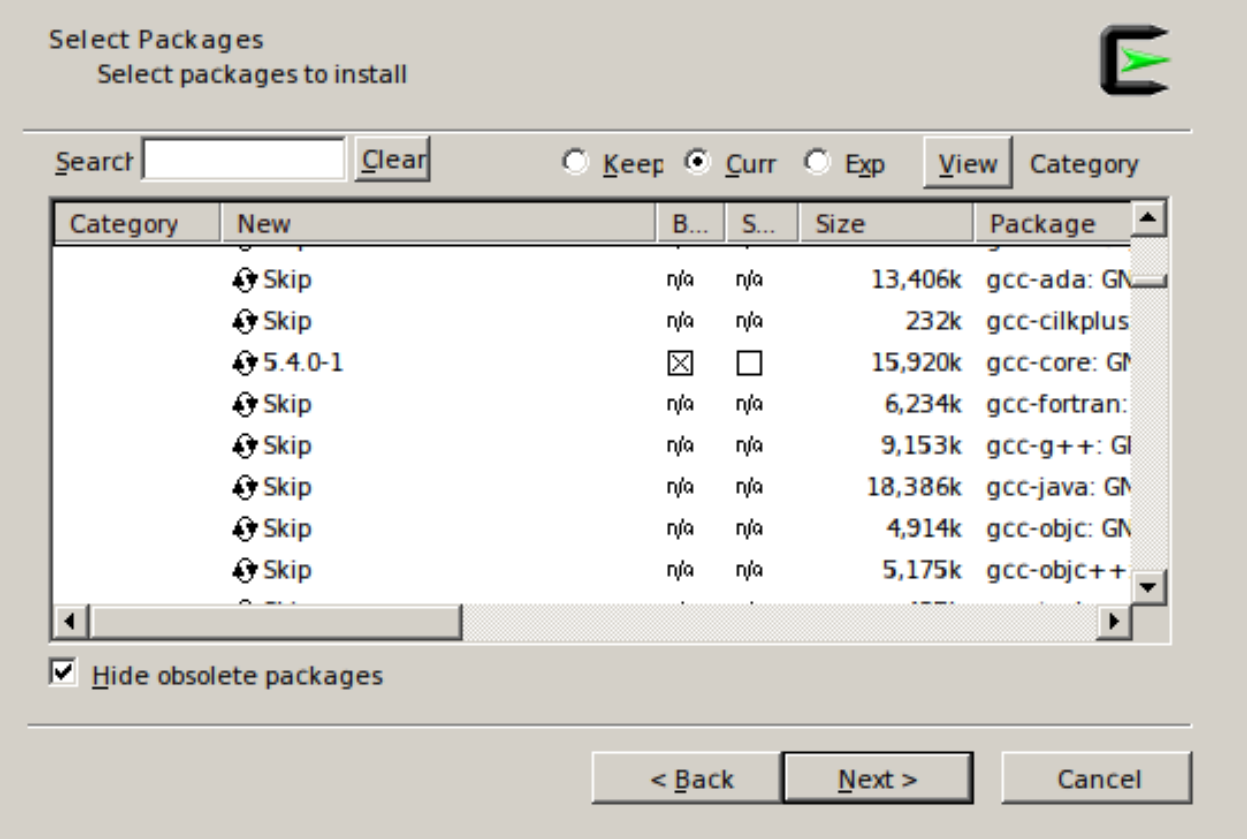
You will need to install Cygwin to get your compiler and tools installed.

First, download Cygwin. If you have a 64-bit computer (you most likely do) download the **`setup-x86_64.exe`** file. Once you have the file downloaded, double click the setup executable program to run the installer.

Step through the installation process. The defaults should be fine until you get to the mirror selection screen. I had good luck picking `http://mirrors.xmission.com`. Once the Cygwin core has been downloaded you will be presented with the Select Packages screen which looks like this:



There are 3 packages under the `Devel` section that you should install: `gcc-core`, `git`, and `make`. Expand the `Devel` section and scroll down to find each of these packages. Click on `Skip` and it will mark the package for installation:



From the **`Doc`** section you should mark the package **`man-pages-posix`** for installation.

Once those 4 packages have been marked for installation, click Next, and then Next again, keeping “Select required packages (RECOMMENDED)” checked.

Once all the packages have been installed, have Cygwin create a desktop shortcut and/or an entry in the start menu. Now run Cygwin and you should be presented with a command prompt that looks something like this:

```
User@ComputerName-d3a3c25ad0 ~

$
```

Type the command **`git`** and press enter. If you see a usage message rather than an error, Git was installed correctly.