How to Install and Use Cygwin

August 28, 2006

Cygwin is a Linux-like environment for someone who's using Windows. It's important to learn how to use Cygwin to compile and run files because you will inevitably have to use Linux, UNIX, or SunOS in the future to do work, thus it's best to get it out of the way now. Also, your coded homework assignments won't be compiled in Windows when we grade them, so it's good to make sure that your code works fine and that you're comfortable in both environments. If at any point during these instructions you get confused or something bizzare happens, call over one of your TAs because we're cracker-jack at fixing problems you may have.

1 Downloading and Installing Cygwin

1. Your laptops came with Cygwin, but not the complete package, the first step is to grab that off their site. So go to http://www.cygwin.com/ and click the "Install or Update now!" link (in the middle of the page, or the side, with the Cygwin icon above it) and download the setup file to somewhere where you can find it.
2. Double click the setup file, click next, and then dot the "Install From Internet" box, then click next again (keep all the settings here the same), then click next one more time.
3. Make sure to set the local package directory to C:\cygwin, then click next, and then next again. Here pick a mirror, any mirror, and then click next again.
4. There's a ton of info on this next page of packages to choose and such, just keep the "Curr" choice selected at the top.
5. In that big long list of packages click the plus sign next to "Editors" to expand that category.
6. Near the bottom of the list of editors under that expansion there's one called "xemacs" with the tagline "A powerful highly customizable open source…". It should be the first one listed right after the "vim" editor, and be 6,731k in size (I only list this because there are several flavours of xemacs, and we want to grab the right one.)

7. Now that we know which editor we're grabbing click the word "Skip" once in the "New" category as we don't want to skip xemacs, we want to install it.
8. Then go ahead and hit next and take a coffee break. Your computer is going to download a plethora of packages, and then install them… (Also, it may tell you that you have to reboot after the install, if it does, do so)
9. Click finish and pat yourself on the back, you've finished a complete Cygwin install.

2 Navigating Around with Cygwin

Cygwin, linux, UNIX, etc… are all primarily command-line based. Thus, in a lot of ways it's more of a pain to deal with (if you've used DOS before it is quite similar). However, you will discover later that command-line based operating systems are actually a whole lot more useful than Windows; that is when it comes to situations where you know what you're doing and want to get something done quickly and efficiently. What follows is a step by step process for setting up a projects directory to do editing / programming in.

1. Run Cygwin.
2. This is the command-line interface, go ahead and type in "cd C:" and hit return.
3. You're now in your root drive on your computer, we're going to make a projects folder so type "mkdir CS1Projects" and hit return.
4. To see what you just did, you can list the contents of the directory by typing "ls" and hitting return.
5. Now type "cd CS1Projects" and hit return to enter your new CS1Projects folder.
6. For a lot more useful UNIX / Linux commands check out http://tinyurl.com/fg4mz
7. And now the real fun begins…

3 Editing Files in Cygwin

There are alot of different editors you can use under Linux / UNIX systems including Emacs, Pico, Nano, and Vi, here we will teach you how to edit using xEmacs because it is the most like Notepad in windows, but feel free to use any UNIX / Linux editor you're comfortable with.

1. While still in your CS1Projects folder type "xemacs helloworld.cpp".
2. You are now in the Emacs text editor, you can move around and type just like in notepad.
3. Go ahead and type in the code from the Lab 1 handout under number 2 on the sheet.
4. Once you've copied the code in order to save and exit Emacs hit Ctrl+X followed by Ctrl+C, then type 'y' to tell it to save the file and exit Emacs. *(Yes we're aware that xemacs has a nice GUI with which you have save, open, and other buttons and menus. However down the line, you'll only be able to use regular emacs which has no buttons, so it's best to learn to open files, edit, save, and exit this way.)*

4 Compiling and Running Code in Cygwin

Unlike with windows, there's no candy-coated GUI to help you out, you must directly tell the Linux / UNIX environment to compile the code, and with what conditions. The advantage of this is that you have much more control and everything works faster than in Windows.

1. Do a quick "ls" to make sure your helloworld.cpp file is in your current directory before we fire up G++.
2. G++ is the GNU compiler for UNIX / Linux systems, right now I'm sure that means little to nothing but keep it in the back of your mind as it will be important later. Type in the following line at the prompt:

   ```
   g++ -o helloworld helloworld.cpp -Wall
   ```

3. Before you hit enter, let's pick through this seemingly jumbled command...
   - The first item, "g++", tells Cygwin to use the G++ compiler
   - The second item, "-o helloworld", is a flag to let G++ know that you want to output the result of this compile to an executable file, a file called "helloworld". If you do not set this parameter it will output by default to "a.exe" (on a true UNIX / Linux system this would be called "a.out" instead).

    - The third item, “helloworld.cpp” is the name of the file to be compiled.
    - The fourth item, “-Wall”, is a flag to let G++ know that you want to know about any warnings that might exist in your code. Warnings won’t stop compilation, but they’re a good indication of sloppy code.

4. If you got any compiler errors or warnings, read them and double check your code before compiling again. Make sure your code matches the code in the lab exactly, as one of the joys of coding is that if you misplace even one semicolon or bracket the code won’t compile. If the problem persists call over a TA.

5. Now you’ve got your compiled executable file, if not then there were some compile errors and you should call one of those oh-so-kind TAs over to help you out.

6. So after all this work, let’s run the thing. However, in UNIX / Linux environments we have to provide a path before running it so instead of typing just “helloworld.exe” and being happy we must include a dot-slash in there, so it’ll be “./helloworld.exe”. You’re always going to have to do this when you try to execute something, if you really want to know why, ask one of the TAs at some point and they’ll explain (this advice goes for about any esoteric question you may have throughout the course, it’s what we’re here for).

7. You should have seen Cygwin output a line of text proclaiming “Hello world. I’m alive!”. If so, you’re golden.

Congrats! You’ve now installed Cygwin, written a program, compiled a program, and learned how to run executables! That wasn’t so hard, was it?. . .