Preparing and Running C Programs for CS 136 (W08)

There are a number of options available to you for developing C code. The choice is up to you. The main thing to keep in mind is that, as in CS 135, the programs you submit for assignments will be tested by us on the CS Undergraduate Environment (CSUE). But, unlike the situation with Scheme, there is no one simple multi-platform solution for C development. Below we list the pros and cons of each approach. You should read them over, then choose a CLE and text editor, and go through any necessary installation procedures.

1 Command-line environments

A command-line environment (CLE) is, as we discuss in lecture, something like an Interactions window for the whole operating system. It provides you with a way to compose complex commands using the keyboard instead of pointing and clicking. You will be editing, compiling and running C programs in a CLE, but there are many ways to do this, and we list the main options for CLEs below, followed by options for text editors. Our recommendations are to use your own Mac or Linux machine if you have one; otherwise either use the lab Macs or use remote access to CSUE. We recommend Cygwin on Windows as a last resort only because of some complexity of installation; once it is installed, it is easy to use.

1.1 Lab Macs

The Macintosh computers in the labs on the second and third floors of MC can be used quite effectively for your work in CS 136. There are two choices for command-line environments. One is to use the Terminal application, found in Applications/Utilities. You can configure the size, background/text colours, and font. This is what will be used for demonstrations in class. The other choice is to use the X11 application, which should be in the Dock. This more general system permits a remote system (such as a CSUE server) to use windows on a local computer. You don't need this capability for CS 136, but it is handy at times, and is necessary if you wish to use the Emacs text editor described below.

Pros: Good support if things go wrong.

Cons: You have to physically be in the labs to use the computers.

1.2 Remote access to CSUE

The servers in the CS Undergraduate Environment are accessible over the Internet. You only need a terminal emulation program. If you are using your own Mac, use Terminal or X11 as described above; for Linux, see the documentation for your particular distribution. In either case, you just type `ssh` followed by the name of a host into the terminal window, and you will be connected to

a server. A host has a name like ~~cpu02~~ linux`.student.cs.uwaterloo.ca`, where the numbers can be `02`, `04`, and so on up to `22`.

If you are using Windows, we recommend the freeware program PuTTY, which you can easily find on the Web. We have a separate document describing how to use PuTTY (not difficult, just a bit of initial configuration required).

Once you have connected to a remote server, you can run the text editors described below in the window you are using, and the files you create will be stored on the remote machine. You can also use `gcc` or `mzscheme` on the command line as described in the slides. When you type `exit`, the connection will close.

Pros: You are using the same machines that will be used to test your programs. There will also be good support if you have problems. The CSUE is used extensively in upper-year CS courses, both major and non-major, and the sooner you get used to it, the better.

Cons: You have to be connected to the Internet for the entire time that you are working. Any disruption of that connection may result in some loss of information.

1.3 Your own Macintosh

If you own a Macintosh computer running Mac OS X, you can use it like a lab Mac. The C compiler is not installed by default, but you can get it from your install disk (under Optional Installs, install the Xcode tools).

Pros: Macs provide you with a good graphical user interface (GUI) and a good CLE, and you can be very productive using them. The environment is quite similar to the CSUE as far as CS 136 is concerned.

Cons: There should be no problems, but if there are any, you will get less support for them if you are using your own machine.

1.4 Linux

If you own a Windows computer, you can run Linux on it. You can download and burn a CD from the Computer Science Club, or you can purchase a CD from them for a dollar or two. Using the CD, you can install Linux on your hard drive, or you can reboot and run Linux from the CD, storing files on a USB flash drive, and leaving your hard drive untouched.

Pros: You get major karma points for running free open-source software. There are many optional packages available for download providing all sorts of added functionality, and extensive user communities active on the Internet. You will find many Linux users in the Math Faculty.

Cons: Installing and configuring Linux can be tricky, especially if you want to keep your Windows installation functioning as before. People at the CSC may volunteer to help you with this, but if problems arise with this or with future use, you may be on your own. If you use the CD, you will

have to reboot to do your CS 136 work, and then reboot again to get Windows running. You will probably have to learn more about Unix than with the other options listed.

1.5 Cygwin

Cygwin is a system that gives you a Unix-like CLE that can be run in its own window within Windows.

Pros: You don't have to reboot, so you can work on CS 136 and do other things at the same time.

Cons: The installation of Cygwin is fairly simple but has several possibly confusing steps. We have a separate document explaining how to do it.

2 Text editors

You will need a text editor within your choice of CLE to prepare your C programs. Again, there are several choices. For those of you using Windows machines, we strongly recommend that you **not** use Wordpad, and under no circumstances attempt to use Word. Our recommendation for all platforms is Vim, with Nano for the cautious and Emacs for the ambitious. Hardcore CS types tend to split between Vim and Emacs and fight with each other over which is better. This is silly; use the one you want, and let others use the one they want. All of the editors listed below are already installed on CSUE and Macs (lab or personal), and can be easily installed on Linux and Cygwin.

2.1 DrScheme

DrScheme has the capability of editing text files without treating them as if they are Scheme programs. It does this automatically for files ending in `.txt`, but your C programs will end with `.c`. Still, you can change the mode of any file edited by DrScheme using the Modes menu item at the bottom of the Edit menu.

Pros: You should already be familiar with DrScheme.

Cons: It will not provide you with any of the support you are used to with Scheme programs (syntax highlighting, automatic indentation, and so on). Most of the other editors will do at least some of this. DrScheme is best used for Scheme programs, not C programs. We recommend that you try to learn how to use one of the other editors listed.

2.2 Nano/Pico

Pico is a simple text editor designed at the University of Washington. Nano is an open-source version of pico developed by the Free Software Foundation. You can consider them as equivalent,

and we will just discuss the more recent Nano here. It is started from within a CLE with the command **nano**, and managed entirely from the keyboard without using a mouse.

Pros: It is very simple to use. At any point, the most likely commands are displayed at the bottom of the screen, including contextual help. When you type, what you type is added to the file you are editing. All commands are given using the "control" key. For example, when you start Nano up, among the commands listed at the bottom is `^X`, which is described as "Exit". That means that if you hold down the "control" key and then press the "x" key, you will leave Nano.

The mail and news reading program **Pine**, which you can use on CSUE to read the course newsgroup, uses the same interface.

Cons: It is perhaps too simple. You can get a certain amount of limited syntax highlighting, but no auto-indentation. Even for that, you have to learn how to configure Nano (by writing commands in a certain format into a certain file). As such, it provides you with basic practice in using a keyboard-oriented text editor, but you may want to move to one of the next two editors.

2.3 Vim

Vim is a modern version of the legendary text editor **vi**, developed for Unix in the 1970s. In contrast to Nano, it is a *modal* editor. In insert mode, just about anything you type is recorded as text; to issue commands, you must switch to command mode, at which point most commands can be issued with one or two keystrokes. You start it by typing **vim** in the CLE (it tells you how to exit when it starts, so pay attention).

Pros: Once you get used to modality, Vim can be very fast to use, especially if you are a good touch typist. It does a certain amount of autoindenting of C code, and if your terminal emulator supports colour (or if you download one of the graphical versions for your platform), it can do syntax highlighting. There is a good tutorial built into the program itself. See also the tips at `http://jmcpherson.org/editing.html`, and useful graphical tutorials and "cheat sheets" at `http://www.viemu.com` (scroll down).

Cons: Since just about every key on the keyboard is used for some command or other, using Vim can be counterintuitive, and you may find it slow at first.

2.4 Emacs

Emacs is a very powerful text editor which is highly reconfigurable through an extension language which is a dialect of Lisp (and so of particular interest to CS 135/136 students). It is not modal in the way Vim is; text is just typed in, while commands are issued using modifier keys such as control, option, and escape.

Pros: You can do just about anything in Emacs, and set it up just the way you like. It has terrific support for programming in many languages. Invoking it under X11 lets you use multiple windows,

colours, and the mouse. It even has a mode that is very close to Vim. Many basic Emacs editing commands also work in Mac applications and in DrScheme.

Cons: Emacs is huge, and you probably won't use 99% of it. The stock criticism of it is "Nice operating system, shame about the text editor". It can take quite a while to get used to it enough to use it efficiently. We recommend that you try it out in CSUE or on the lab Macs before committing to it.

3 C compiler

The C compiler, `gcc`, is installed on CSUE and the lab Macs. If you have your own Mac, you can install the Xcode tools (under Optional Installs on your install disk) to get it. All versions of Linux either pre-install it or let you install it over the Internet. It is an optional install for Cygwin as described in more detail in the Cygwin document we have prepared.

4 Running mzscheme

You can run `mzscheme` on the command line in CSUE (and probably on Linux) just as described in the slides. For the other systems, you have to know where it is. For Macs, it is located in the `bin` directory within your PLT Scheme installation in the `Applications` folder. For Cygwin, it is probably in `/cygdrive/c/Program Files/PLT`.

That's a lot of additional typing, which you can save by changing the search path for commands to include the directory containing `mzscheme`. To do this, edit the file `.bashrc` in your home directory, and add the line "`export PATH=<added>:$PATH`" to it, where `<added>` is the directory containing `mzscheme`. If there are any spaces in the directory path (as with the space between `Program` and `Files` in the Cygwin example above) you will have to precede them with a backslash.