

UNIVERSITY OF THE WEST INDIES
Department of Mathematics and Computer Science
CS20A—Information Structures and Algorithms
Semester II, 2004
Lecturer: Dr. Daniel Coore

Lab Guide

Guide to Using the Lab

Introduction

This guide should enable you to use the tools provided in the lab for doing your CS20A assignments and projects. It is still under development, so it is still quite sketchy, but hopefully enough to allow you to do your assignments and projects.

Notation

Emacs uses the Control and Meta (probably Alt on the lab keyboards) key modifiers quite extensively to get more functionality out of the keyboard. This approach to building an editor can prove to be very effective for rapidly entering data because minimal mouse movement is required. Nevertheless, Emacs does provide you with a menu bar which does react to the mouse, so if you are accustomed to working with menus in this way, you should still be able to operate efficiently using Emacs.

Whenever we want to indicate that you should press and hold down the Control key, we prefix the key with **C-**. A prefix of **M-** means that you should press and hold down the Meta (Alt) key. When both prefixes are present, you should press and hold down both the Control and the Meta keys together in addition to the other key indicated. For example, **C-M-f** means press and hold Control and the Meta key while typing the letter **f**.

Using Scheme in the Lab

The first thing you will need to do is get a usable Scheme editing environment up and running. You can accomplish this with the following steps:

1. Login
2. Download the project code from the course web page (if any has been provided). Save and unzip it in a subdirectory of your home directory.
3. Get a shell, and type at the prompt **schemacs &**

If all goes well, Emacs should launch, and automatically start MITScheme under the interactive control of a buffer called ***scheme***. Use this buffer for typing your interactive instructions, e.g. code tests and debugging. To evaluate an expression, type Control-x Control-e (written **C-x C-e**) at the end of it. For example to evaluate $(+ 1 2)$, you would type in your ***scheme*** buffer:

(+ 1 2)C-x C-e

You should see the result printed beneath your expression, at which point you can type some more expressions to be evaluated.

For typing bigger pieces of code to be evaluated by the Scheme interpreter, you should use a file. To open a file within Emacs, type Control-x then Control-f (written C-x C-f). The cursor will appear at the bottom of the emacs window, and you may now type the name of the file you would like to load. Emacs will do completions automatically for you, if it can, press the TAB key after typing the first few characters of the file you want to load, and Emacs will complete the name as far as it can, if it hesitates before showing the complete filename you want, then it means there is more than one file with that beginning, and you should type the next character or two and press TAB again (of course, you could just type out the entire filename). When the filename is correct, press ENTER to load it. If the filename ends in `.scm`, the buffer will automatically be in Scheme mode.

Common Commands

While editing Scheme code, there are some common editing features that you will want to perform. Emacs already shows you which parentheses are being balanced when you close them, but you will also want to be able to move the cursor around characters, words and expressions. To move the cursor back one character, type C-b, for forwards one character type C-f. For those commands, you can also use the arrow keys. To move back or forward a whole word, type M-b or M-f, and to move back or forward a whole expression, type C-M-b and C-M-f. [A table listing other common commands will be presented soon]. Two other keystrokes that are very useful while developing Scheme code are M-p and M-n. After evaluating an expression, (and typing C-x C-e), try typing M-p, you should see your most recent evaluated expression appear, ready for re-evaluating. If you press M-p again without any other intervening keystrokes then you'll see the second most recent expression evaluated. You can continue this all the way back to the first expression you evaluated (provided you haven't been using the Scheme interpreter for so long that Emacs has forgotten). If while cycling back in search of a particular expression, you accidentally pass the one you want, pressing M-n will move forward in your history. Both M-p and M-n can prove to be very handy while interactively debugging code.

In any mode in Emacs (including Scheme mode), you can get a list of the keystrokes that have special meaning by typing C-h m. Use this to get a list of some other commands for now until this document can be further updated.