

CS51 Guide to Emacs

CS51 — Spring, 2006

January 28, 2006

1 Introduction

Using a good text editor can save you a significant amount of time by giving you

- Syntax highlighting to help find syntax errors
- Automatic indentation to make your code readable
- Support for compiling and testing program within editor
- Ability to perform sophisticated text manipulations such as regular expression searches and replace operations

We encourage you to use Emacs in CS 51 because it works well for programming in both Scheme and C++. In either case, you will likely find that Emacs's ability to display multiple windows crucial.

2 Basic Setup

You'll certainly want to turn on syntax highlighting—syntax highlighting catches many errors before you even try to run your code. For instance, if you have an extra parenthesis—`(car (cdr '(a b c)))`—Emacs will color it purple. You will find this syntax highlighting even more significant when we reach C++.

To turn on Syntax highlighting, you have two choices. If you're running the graphical version of Emacs over X11—for setup, see <http://www.fas.harvard.edu/lib50/Reference/emacs>—you can simply go to the **Options** menu and select **Syntax Highlighting (Global Font Lock mode)**.

You may also made this change manually by editing your `~/.emacs` file¹ and adding in the following line: `(global-font-lock-mode)`.

3 Running Emacs

To run emacs, simply type `emacs` at the shell. This causes Emacs to load in the foreground of the shell. If you're using X11, you run Emacs in the background and keep control of the shell by running Emacs like this: `emacs &`.

You can also specify the name of a file to edit or create as an argument to the program: `emacs ~/.emacs &`

¹The `~` indicates that the directory path begins with your home directly. The dot preceding `emacs` indicates that it is a hidden file. To see it, you must use the `-a` flag to `ls`. You can simply edit it, however, by typing `emacs ~/.emacs` from the shell.

3.1 The Meta Key

Emacs relies heavily on two keys: the control key and the meta key. The meta key is usually the alt key on a Windows or Linux machine, and the escape key on a Mac.

Meta and Control are frequently abbreviated as simply **M** and **C**. This document follows the same convention.

4 Scheme

4.1 Running the Interpreter in Emacs

You can run Scheme directly in side Emacs simply by typing **M-x run-scheme**; this will create a buffer called `*scheme*` with the MzScheme interpreter.

From the interpreter, you can quickly load your entire file by typing `(load "filename.scm")`. This will evaluate every expression in the file, possibly including any tests you add to the end of the file.

You can also scroll through the command history in the Scheme interpreter by pressing **C-`<up arrow>`** and **C-`<down arrow>`** to move up and down, respectively.

4.2 Using Two Windows

You can create two screens in your emacs window by typing **C-x 2**. You can switch (move your cursor) between the two screens using **C-o**. When in a screen you can switch between buffers using **C-x b** or open a new file with **C-x C-f** or kill a buffer with **C-x k**.

4.3 Searching and Replacing

M-x query-replace allows you to type in the word; then it prompts you for the replacement. The cursor jumps to each match; hit **space** to replace the word and **n** to skip the word. The search begins from the initial position of the cursor in the buffer and continues to the end of the buffer.

4.4 Interpreting Scheme inside Emacs

Once you've started the Scheme interpreter in Emacs, you can have it evaluate Scheme expressions using the interpreter. So long as you are in a file with the correct extension, `.scm`, if you have a single expression you want to test, you can use the sequence **C-x C-e** at the last parenthesis of the expression to evaluate it. The result will appear in the buffer containing the Scheme interpreter.

This leads to a useful trick for testing code: you can write your test cases inside of comments—`|#` and `#!`—load your file, and individually evaluate the test cases separately.

5 C++

Emacs is also an effective C++ editor. When programming in C++, you can compile by typing **M-x make** and hitting enter. After hitting enter, you may optionally specify a target for the makefile: **M-x make**, hitting enter, and typing the name of the target, and hitting enter again. Running make will bring up a new buffer showing you the result.

5.1 GDB in Emacs

Emacs also integrates with GDB to allow you to look at your code while stepping through your program. To run GDB in Emacs, type `M-X gdb <executable>`, where `<executable>` is the name of your program. If you have a core file, you can open it with `M-X gdb <executable> <core>` where `<core>` is the name of your core file.

When debugging your program, Emacs will place an arrow next to the line of code about to be executed.

When working with GDB in Emacs, `C-↑` and `C-↓` work just as they do in the Scheme interpreter.

6 Useful Shortcuts

<code>C-g</code>	Cancel whatever action is currently happening
<code>C-x C-f</code>	Open a file or create a file
<code>C-y</code> or <code>C-y</code>	Paste selected region.
<code>C-k</code>	Cut text to the end of a line.
<code>C-X k</code>	Kills a buffer, defaulting to the current buffer
<code>C-X 2</code>	Force Emacs to split the screen across two buffers
<code>C-X b</code>	Switch buffers
<code>C-p</code>	Move cursor up
<code>C-n</code>	Move cursor down

For many more useful keyboard shortcuts, see the CS 50 Emacs Reference Card (<http://www.fas.harvard.edu/lib50/Reference/emacs/refcard.pdf>).