

How to use SML in Emacs

CMPU-235: Programming Languages

Vassar College, Fall 2017

1	Overview	1
2	Using the Department Labs	1
3	Emacs Installation	2
4	Emacs Basics	2
5	SML-mode for Emacs installation	5
6	Using the SML/NJ REPL (Read-Eval-Print Loop) in Emacs	6
7	Acknowledgments	8

1 Overview

We have been working with the Standard ML programming language, using the SML/NJ (Standard ML of New Jersey) compiler. You will need SML/NJ and a text editor on your computer to do the programming assignments. Any editor that can handle plain text will work, but I recommend using Emacs and its SML Mode, which provides good syntax highlighting, indentation, and integration with an SML environment. While Emacs does not have the look-and-feel or tool-integration of many modern integrated development environments (IDEs), it is a versatile tool well-known by many computer scientists and software developers.

This document describes how to install, configure, and use Emacs, SML/NJ, and SML-Mode-for-emacs (henceforth SML Mode) on your computer. These instructions should work for recent versions of Windows, Mac OS X, and Linux.

We also describe how to use it on the machines in the department labs, where most of what you need is already installed on the Linux machines.

2 Using the Department Labs

If you use the department Linux machines, then the recent versions of Emacs and SML/NJ has already been installed for you, allowing you to skip Sections 3, 5, and 7 below. So here is all you need to do:

1. Open Emacs. On Linux, open an xterm (X terminal) window and type `emacs`. The `emacs` binary is located at `/usr/bin/emacs`.
2. Read Section 4 to get familiar with Emacs.
3. Follow the instructions in Section 6 to install SML Mode for Emacs. You should need to do this only once.
4. Read Section 8.

3 Emacs Installation

(We strongly recommend Emacs version 24.X (for any X) so that you can use the most recent version of SML Mode. Earlier versions of SML Mode are fine, but they are more difficult to install. You can check the version of an Emacs installation in several ways, including the “About Emacs” option under the “Help” menu. Installing version 24 is easy, so we recommend doing so even if you already have an older version.)

Directions depend on your operating system:

3.1 Windows

Download a zip archive of the most recent full version, currently Version 24.5, available at <http://ftp.gnu.org/gnu/emacs/windows/emacs-24.5-bin-i686-mingw32.zip>. (More information and other versions of Emacs are available at the GNU Emacs website, <http://www.gnu.org/s/emacs/>.)

Unpack the downloaded zip archive file [emacs-24.5-bin-i386.zip](#) by right-clicking it and choosing *Extract All*. This should produce a folder called [emacs-24.5-bin-i386](#). Open it to find a single folder named [emacs-24.5](#). Move this [emacs-24.5](#) wherever you want, but pick a permanent place (i.e., do not move it again after the next step).

Once you have moved the [emacs-24.5](#) folder to where you want, look inside to find [bin\addpm.exe](#) (the [.exe](#) extension might not be visible in the folder window depending on how you have Windows configured). Double click on this file to run it one time and it should add a *Gnu Emacs* folder to your *Start* menu and do some other setup operations. Open the *Start* menu and select *Gnu Emacs* → *Emacs* (or just type *emacs* in the search box) to launch Emacs.

3.2 Mac OS

Download Emacs as a Mac OS X application from <http://emacsformacosx.com/>. Open the disk image file ([.dmg](#)) and drag the Emacs application to your Applications folder. If you prefer another version of Emacs, such as the more primitive one on the command line or Aquamacs (<http://aquamacs.org/>), you can use it, but make sure it is based on a version of Emacs 24X or newer.

3.3 Linux

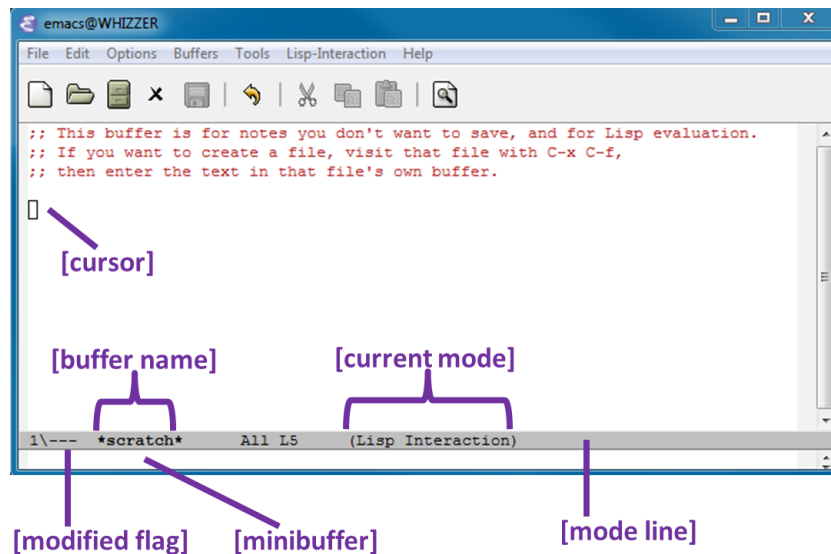
Emacs is probably already installed. If not, use your package manager to install it. On Ubuntu and other Debian derivatives, try `sudo apt-get install emacs`. On Fedora, try `sudo yum install emacs`.

4 Emacs Basics

Using Emacs feels a little different than using other editors; it can take some getting used to, especially the keyboard shortcuts. Fortunately, Emacs has buttons and menus to help you adjust if that is your style. The following is a short primer on Emacs terminology and keyboard commands.

Beyond the basics described here, which should be all you need, there are countless free resources available. An introductory “tour” is at <http://www.gnu.org/software/emacs/tour/>. For more information, see the Emacs Reference Manual at http://www.gnu.org/software/emacs/manual/html_node/emacs/index.html (also available within Emacs from the Help Menu), the Emacs Wiki at <http://www.emacswiki.org/>, or the Emacs Tutorial (within Emacs from the Help Menu). In case you prefer watching a 10 minute screencast introduction to Emacs, there is one available at https://youtu.be/K3_SFerNIeI. If that intrigues you, you could watch a longer, 40 minute version at <https://youtu.be/B6jfrwR10k>. And, if that makes you really interested in Emacs, you can take a look at a Youtube channel called “Emacs Rocks” at

<https://www.youtube.com/user/emacsrocks> which showcases some really nice features of Emacs in reasonably small chunks. I found these screencasts very useful—even when I watched it after having used Emacs for more than 15 years. I learned a few new tools and generally, really enjoyed watching it.



- The cursor is a rectangular block and is referred to as the **point**.
- The **mode line** displays information about the **buffer** displayed in the current **window**. A buffer is a logical “thing” that you are working on. When you open a file, it will be loaded into a buffer, typically with the same name as the file.
- Every buffer is edited in a **mode**. The most basic mode is **Fundamental**, which provides only the most basic Emacs editing features. There are modes for many different programming languages.
- There are many “special” buffers that do not correspond to loaded files. The one above is called ***scratch***. This buffer runs in Lisp Interaction mode, which means that you can interactively type and evaluate expressions in the Lisp programming language. But we won’t do that.

Emacs uses many key combinations involving the **Control** and **Meta** keys. Such key combinations are denoted **C-x** (Control-x (lowercase)) or **M-x** (Meta-x). On keyboards that don’t have Meta (just about all keyboards today), try **Alt** on PC keyboards or the funny-symbol-with-four-circles or **Option** keys on a Mac keyboard. (Emacs might complain about “Super” if you get the wrong one. If you or Emacs gets confused about what you are trying to type, use **C-g** to cancel your current command and start fresh.) If none of these work, use **Esc**, but when trying to type M-x, for example, type Esc *then* type x. (This is only if using Esc as Meta. The other Meta “substitutes” work as usual: hold while pressing the second key.)

A sequence of key presses is written like **C-a C-b M-x**, which would mean do the three actions in sequence: Control and lower-case a, then Control and lower-case b, then Meta and lower-case x.

4.1 The most important commands in Emacs

- **C-x C-c** Emacs
- **C-g** the current action
- **C-x C-f** Open a file (whether or not it already exists)
- **C-x C-s** Save a file
- **C-x C-w** Write a file (probably more familiar to you as *Save as...*)

4.2 Cut, copy, paste

- Highlight text with the mouse or by hitting **C-Space** to *set a mark* and then moving the cursor to highlight a region.
- **C-w** Cut a highlighted region
- **M-w** Copy a highlighted region
- **C-k** Cut (*kill*) from the cursor to the end of the line
- **C-y** Paste (*yank*)

4.3 Some other useful commands

- **C-x 2** Split the window into 2 buffers, one above the other (Use the mouse or **C-x o** to switch between them)
- **C-x 3** Split the window into 2 buffers, one next to the other (Use the mouse or **C-x o** to switch between them). You can use this and the earlier command more than once to create a tiled workspace.
- **C-x 0** Remove the *region* (or, the buffer) that includes the cursor so there is less buffer.
- **C-x b** Switch to another buffer by entering its name
- **C-x C-b** See a list of all current buffers

4.4 Getting help within Emacs

In addition to the help button/menu on the right...

- **C-h** Help. Hitting this will display a short message in the minibuffer: **C-h** (Type ? for further options).
- **C-h b** Key *bindings*. This lists all key bindings that are valid for the current mode. Note that key bindings change from mode to mode.
- **C-h a** Command *apropos*. After typing **C-h a** you can type a symbol and a window will appear that lists all symbols and functions that match that phrase.

4.5 More advanced Emacs hacks

If you are curious, try some of these once you have finished the rest of the setup instructions. They are unnecessary for any of the work we will do, but may be convenient.

- Change the colors of your syntax highlighting. In version 24 of Emacs, M-x customize-themes is a good place to start.
- General customization interface: Open the *Options* menu and choose the first item under *Customize Emacs*. This will let you customize Emacs through a sort-of-graphical interface. It saves all your settings in a file in your “home” directory, `~/.emacs`.
- Much, much, much more: Emacs calls itself an *extensible* editor for a reason.

5 SML-mode for Emacs installation

SML Mode is an extension to Emacs that is not Emacs itself or SML/NJ itself. It displays SML code nicely with syntax coloring and clean indentation, and provides a way to run SML from within Emacs.¹

To install the current version of SML Mode (currently 6.7), follow these instructions from within Emacs:

1. Run the command M-x `list-packages` (and then *Return/Enter*). If the `list-packages` command does not exist, your Emacs version is too old. You can upgrade Emacs or follow more difficult SML Mode installation instructions below.
2. Find `sml-mode` and click on it with your mouse. If you do not see it or installing it does not seem to work, see below.
3. Click on `install` with your mouse.
4. Exit and restart Emacs.
5. Read below to see if you need to follow a couple more steps (more likely under Mac and Linux).

5.1 Troubleshooting

Use this information only if the automatic process does not work. If you could not find `sml-mode`, first note that while package names are mostly alphabetized, they may be in more than one group, making it seem like `sml-mode` is not present. Check the entire buffer. You can most easily search using C-s in Emacs. If you still do not see `sml-mode`, try killing the buffer (C-x k) and trying the previous step again (some users have reported having to try several times, frustratingly). If you still do not see SML Mode, then you can follow these more manual steps instead:

1. Visit <http://elpa.gnu.org/packages/sml-mode.html>.
2. Locate, and download the latest version (currently `sml-mode-6.7.el`) from that page.
3. In Emacs type M-x `package-install-file` ENTER.
4. At the prompt *Package file name:*, give the path to the just downloaded `sml-mode-6.7.el`, and type ENTER. This will split the window, and show the `*Compile-log*` with some lines about Compiling file, and perhaps a warning. If there are no errors, `sml-mode` should now be installed.

¹Thanks to Stefan Monnier for maintaining SML Mode. The website is <http://www.iro.umontreal.ca/~monnier/elisp/>, but you do *not* need to go there to install SML Mode.

To verify that SML Mode is properly installed, let us check that it does indentation/coloring for SML files and that you can create the SML read-eval-print-loop (REPL) from within Emacs.

First, edit an existing or new SML file (try `C-x C-f test.sml` to create a new file if nothing else is handy). You should see the mode display at the bottom of the Emacs window change from Fundamental (or whatever it was) to SML. If you enter a line of code like `val n = 1;` you should see colors highlighting the keywords and variable names. When you are editing code, whenever you hit the `Tab` key, Emacs will try to reindent the current line appropriately.

Second, while the cursor is in an SML buffer (i.e., you are editing an SML file), run `C-c C-s`. This should split the window and create an SML prompt in a new buffer. In that buffer, you should be able to type `1+1;` at the prompt and see 2 as the result.

If you are seeing syntax highlighting, but the `C-c C-s` command fails with an error message, Emacs is probably having trouble finding the SML program. You can hopefully fix this as follows:

5.2 Mac OS

In Emacs, edit your `.emacs` file by `C-x C-f ~/.emacs` (that is tilde, slash, dot, emacs) to open the file. Paste in these lines:

```
(setenv "PATH" (concat "/usr/local/smlnj/bin:" (getenv "PATH")))
(setq exec-path (cons "/usr/local/smlnj/bin" exec-path))
```

Save the file (`C-x C-s`). Exit and restart Emacs.

5.3 Linux

Find where `smlnj-110.81` was installed. Then follow the Mac OS instructions above, but replacing `/usr/local` with the appropriate path.

6 Using the SML/NJ REPL (Read-Eval-Print Loop) in Emacs

At this point, we are done installing! This section shows you how to run SML programs from within Emacs. It assumes you already have an SML file or can write your own SML program in a new one.

1. Edit a file with extension `.sml`. You should be in SML-mode, using `Tab` to indent your code well.
2. To create the `*sml*` buffer (which holds the REPL), type `C-c C-s` (and then *Return/Enter*) in the buffer with the `.sml` file. (Note: This will not work in the `*scratch*` buffer that Emacs starts in because this buffer is not in SML Mode.)
3. Keep the `.sml` file(s) you are working with for a particular assignment in the same folder. When you type `C-c C-s` to start the REPL from a buffer for `foo.sml`, the REPL will look in the right folder for `foo.sml` when you type `use "foo.sml"` and will look in the same folder for any other file you use such as `foo_tests.sml`. This is less confusing than trying to keep track of different folders and paths while using the REPL although that is possible.
4. To end and restart a REPL session, type `C-d` (to end it) and `C-c C-s` (and then *Return/Enter*) (to restart it). You must type `C-d` while in the `*sml*` buffer; you can type `C-c C-s` from the `*sml*` buffer or a buffer with a `.sml` file.

5. By ending and restarting a session, the new session has an empty environment. Your earlier interactions are still in the `*sml*` buffer, so you can save them, cut-paste them, etc., but they have no effect on the evaluation in the restarted REPL session.
6. Evaluation can go into an infinite loop.
 - This has likely occurred if you are not getting the “- -” prompt back and nothing appears to be happening.
 - `C-c C-c` will interrupt evaluation and get you your prompt back.
7. If you forget to end your binding with a “;” character, the REPL will print an “=” character on the next line, which is just its way of saying, “you are not done – continue your binding,” so type a “;” and hit *Return/Enter*. This is not an infinite loop (nothing is being evaluated; the REPL is waiting for you) so `C-c C-c` does not do anything.
8. If the printed result looks “pretty good,” but part of what you expected to see has been replaced by a “#” or “...,” do not worry. The REPL has a limit on how many characters it prints, which is good since you might make a large value, such as a list with tens of thousands of elements. You can adjust the limit if you want.

6.1 Some advice (You will be glad you followed!)

In each REPL session, follow this pattern:

- First type `use "foo.sml";` for any SML files you want to use.
- Then use the REPL manually as long as you wish.
- After using the REPL to test something, do *not* use `use` to load (or reload) any more files.
- When tempted to violate the previous point, end and restart your REPL session before continuing.

Why: `use "foo.sml"` has a very simple semantics: it adds the bindings in the file to the environment in order. These may or may not shadow bindings from the last time you typed `use "foo.sml"`, depending on how `foo.sml` changed. This confuses even expert programmers until they train themselves to follow the pattern above.

If you find yourself typing the same non-trivial things repeatedly in the REPL, stop wasting your time.

- Move the repeated parts to a second file, e.g., `test.sml`.
- Then, when you restart your session, begin with `use "foo.sml"; use "test.sml";`.
- In fact, there is an even faster way:
 - Begin `test.sml` with the expression `use "foo.sml";`
 - Then begin your session with `use "test.sml";`

Note: Do *not* put `use "foo.sml"` in `test.sml` and begin your session with `use "foo.sml"; use "test.sml";`. That will evaluate the bindings in `foo.sml` twice, which is confusing.

If you develop some emotional attachment to the transcript of your `*sml*` buffer, you can save it to a file just like any other buffer. But after you do, it's not an `*sml*` buffer anymore, so you will have to create a new `*sml*` buffer from a buffer in SML Mode via `C-c C-s`.

7 Acknowledgments

These instructions were prepared starting with material created by Ben Wood, adapted from prior materials by Dan Grossman and Hal Perkins. This was further modified by Sameer Pradhan. The changes included, among other things, addition of a few more keyboard commands, pointers to Youtube tutorials and talks on Emacs. Stefan Monnier provided fantastic feedback on this document and even created SML Mode version 6 to simplify SML Mode installation substantially.