SML/NJ CM Usage

Last Revised: 31st August 2011

Managing code that lives in several SML files—loading the files in the correct order and avoiding shadowing issues as the program grows—becomes irritating quickly. In order to avoid tedious and error-prone sequences of `use` commands, the authors of SML/NJ included a program that will load and compile the source code for programs from a text file that lists the files which contain it.

The structure `CM` has a function

```
val make: string -> unit
```

`make` takes the name of the listing file, a file usually named `sources.cm`, that has the following form:

```
Group is

$/basis.sml
file1.sml
file2.sml
file3.sml
```

Note that `CM` will not run if any of the files specified is empty or contains no module definitions, so you may occasionally need to add files to `sources.cm`, or uncomment lines that we include.

Also note that while `CM` does compile all the code found in the files specified, it only introduces bindings into the top level-environment for values, types, and exceptions defined within modules. All other bindings are discarded.

Once you have `sources.cm` set up, loading your code using the read-eval-print-loop (REPL) is simple. Launch SML in the directory containing your code, and call `CM.make "sources.cm"`:

```
$ sml
Standard ML of New Jersey v110.69 [built: Wed Apr 29 12:25:34 2009]
- CM.make "sources.cm";
[autoloading]
[library $smlnj/cm/cm.cm is stable]
[library $smlnj/internal/cm-sig-lib.cm is stable]
...
-
```

Call `CM.make "sources.cm"` again when you've changed something in your code and want to recompile.

The compilation manager offers a better interface to the command line. There is less typing and less of an issue with name-shadowing between iterations of your code. In short, when using `CM` your development cycle will be:

1. Edit your source files.
2. At the REPL, type

   ```
   CM.make "sources.cm";
   ```

3. Fix errors and debug.
4. If done: celebrate! If not: go to step 1.

Calling `CM.make` will make a subdirectory in the current working directory called `.cm`. This is populated with metadata needed to work out compilation dependencies, but can become quite large. Since it's entirely generated, the `.cm` directory can safely be deleted between work sessions.

An extremely detailed discussion of CM and its implementation in SML/NJ can be found at `http://www.smlnj.org/doc/CM/new.pdf`.