Sample Project

I’ve put together a sample project to illustrate a suggested physical architecture and Makefile system. I do not expect or suggest that you look at the code, just at the way the files are laid out and at the way the Makefiles are set up.

- Check it out:
  `cvs -d /home/barrett/public/repository co sample_project`

- Browse it online:
  Click on Sample Project from the course web page.

To build the sample project on a CIMS machine, edit `Makefile.local` and change the `TOP` variable to point to your local working copy.

Then type `make`.

Sample Makefile System

To use the Makefile system from the sample project for your own code, you must do the following:

- Mimic the directory structure of the sample project:
  `project`, `project/bin`, `project/src`,
  `project/src/include`, `project/src/main`,
  `project/src/module1`, `project/src/module2`, ...

- Copy files from the sample project: `Makefile`, `Makefile.local`,
  `Makefile.std`, `bin/libmerge`, `src/Makefile`

- Edit `Makefile.local` and change the options as appropriate for your project

- Edit `src/Makefile` to contain your header files and modules

- Create a Makefile for each module similar to the ones in the sample project (note the difference between a library module such as `expression` and an executable module such as `main`).

Sample Makefile System

What can you do with this Makefile system?

- Easy to add new source files. Module files are listed in
  `src/module/Makefile`. Public header files (in `src/include`) are
  listed in `src/Makefile`. New modules must be added to module list in
  `src/Makefile`

- Support for multiple platforms. To compile for a different platform (i.e. object files, libraries, and executables are stored separately), change the
  `PLATFORM` variable in `Makefile.local` or type `make PLATFORM=newPlatform`

- Support for multiple build-types. Default build is debug. To compile the
  optimized version, uncomment the `OPTIMIZED=1` line in
  `Makefile.local` or type `make OPTIMIZED=1`. 
Support for unit testing:
1. Create test directory `src/module/test`
2. Write unit test in `src/module/test/test.cpp`
3. Create local `src/module/test/Makefile` (see example in `sample_project/src/expression/test`)
4. Type `make` in `src/module/test` to build unit test

Support for source searching in `emacs`
- To find a class declaration:
  1. Move the cursor onto the class name
  2. Type `M-`
  3. Make sure `emacs` has the right class name and press return
  4. Give path to tags table: `project/` and press return
  5. To go back to where you were, use `M-x b`
- To search through all files in your project
  1. Type `M-x tags-search`
  2. Enter the expression to search for
  3. If you haven't before, give the path to the tags table: `project/`
  4. Use `M-`, to “find next”