

CS350: Notes on using the tiger project within Eclipse

Compiling. *Compilation should work, but “Automatic Build” may need to be off and you may need to use “Project → Clean”.* As far as I can tell, the “CDT” system for doing C/C++ programming does not understand how to use compiler construction tools (e.g., `flex` and `bison`). However, it does provide a way to let me configure the compilation as a “Standard make” project rather than a “Managed make” project. Unfortunately, “Automatic Build” seems to work only sometimes for this project. If you have trouble with it (or prefer not to use it), just turn it off and choose “Build Project” on the “Project” menu to build it. If it doesn’t build when you do this, select “Clean” from the “Project” menu. Note that the project will be built automatically when the tests are run — but this won’t tell Eclipse how to highlight your errors in the program source. **WARNING:** do not have Eclipse and `Run-Tests` perform builds simultaneously — I suspect you will get something inconsistent if you do (but cleaning via “Clean” under the “Project” menu should fix it).

Running. *You’ll need to indicate “end-of-file” when you run your program.* I don’t know how to tell Eclipse to communicate an “end-of-file” when typing input into a running program, so there’s no way to terminate your lexical scanner/parser (and thus no way to run most of the later steps of the compiler) when you run tiger from Eclipse. You can deal with this by (a) starting a terminal window and running tiger from the command line and using control-d to indicate end-of-file, (b) always putting your test program in a file and giving tiger that file name, or (c) adding a special token that the lexical scanner will recognize and treat as end-of-file (this should be something that won’t appear in a correct tiger program, such as `@EOF@` or `^d` or something).

By the way, to get it to run on the Macintosh version of Eclipse, I had to change the “binary parser”, by going to “Properties” in the “Project” menu, selecting “C/C++ Make Project”, clicking on Binary Parser, and turning on Mach-O parser. This should already be taken care of in your project when you get it.

Debugging. *This should already work when you get the project.* To be able to debug, I had to select the debugger manually to keep it from saying “no such debugger” — I did this in the “Debug...” menu, under the “Debugger” tab, by choosing “gdb” from the “Debugger” pull-down menu there.

Testing. *The following should make it easy to run the tests:* We’ll be using the `Run-Tests` script for testing throughout the semester, so you may want to configure the “External Tools” menu to let you run it easily: Select “External Tools” from the “External Tools” submenu of the Eclipse “Run” menu; then select “Program” and click on the “new” button; By “Name” enter `Run Tests` (or anything else you like); Under “Location” enter `${project_loc}/Run-Tests`, and under “Working Directory” enter `${project_loc}`. Finally, click on the “Apply” button and then the “Run” button at the bottom. You should then be able to run the tests by clicking on the external tools icon (the one with the little red suitcase).

Contributing Tests. When you come up with an interesting test, i.e. one that that does (or could) break your program in a way not demonstrated by the other tests, please put it in the `added` subdirectory of the shared tests directory (found in the `cs350/Examples` directory in your home folder). See the `README` file there for details.