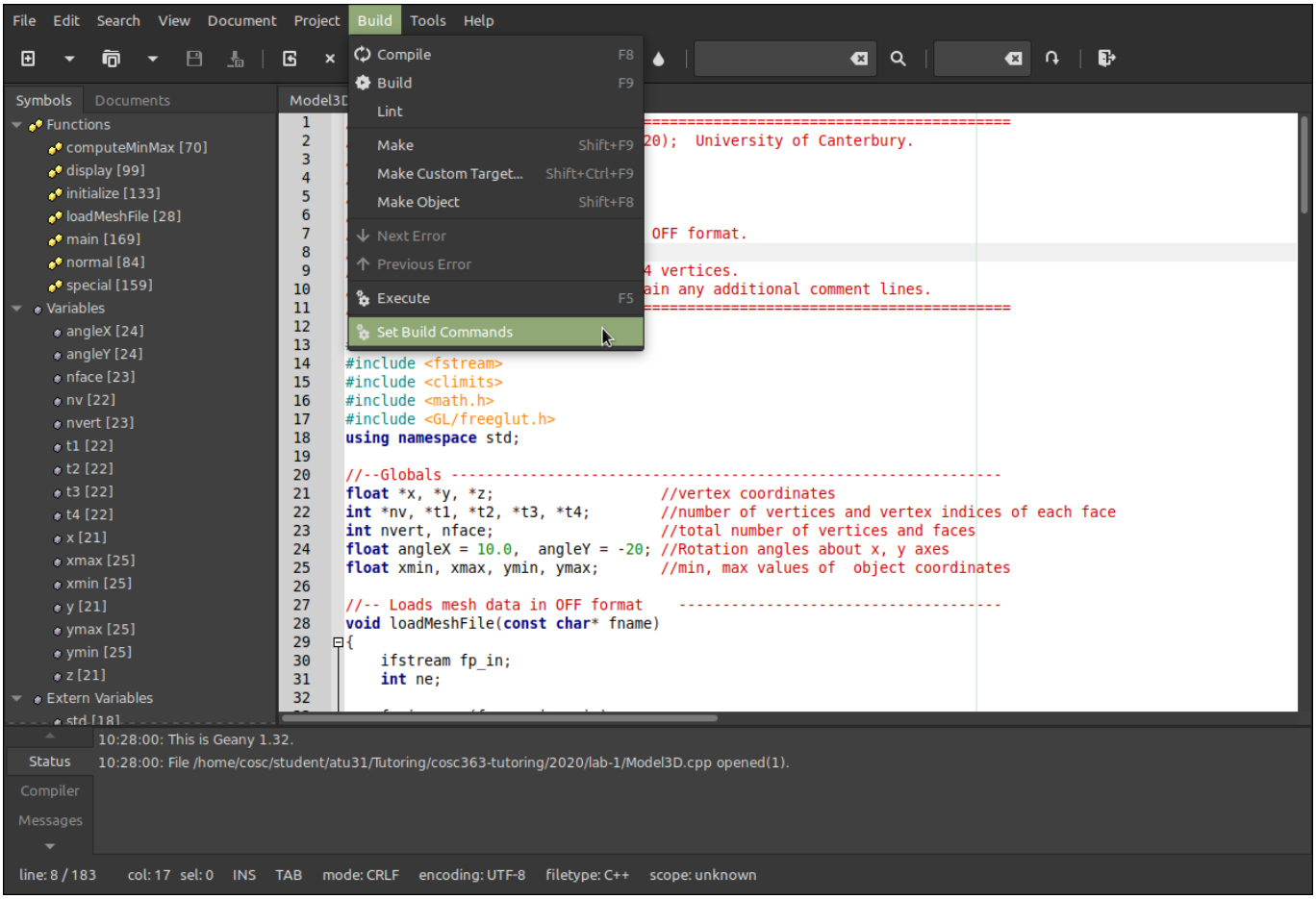
Compiling COSC363 Labs

All COSC363 labs use the C++ programming language. Since C++ is a compiled language your code must be compiled before your program can run. We will be using the g++ compiler in the labs

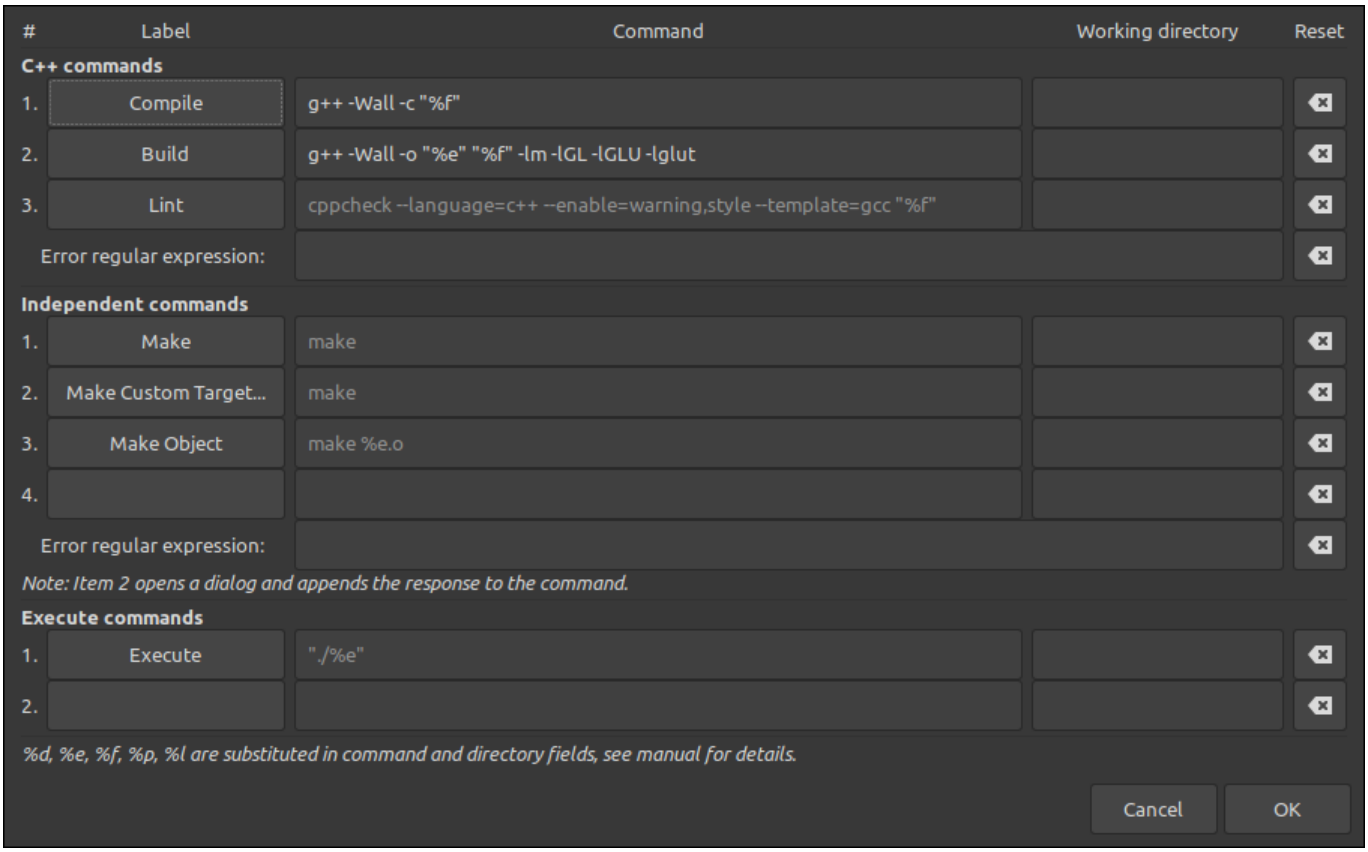
Setting Up Geany

Geany can also be configured to compile the labs by modifying the build command to include the necessary libraries. Access the build settings using the "Set Build Commands" window under the "Build" tab.



For most labs your Geany build command ("C++ commands" > "Build" in the "Set Build Commands" dialog) should look like this:

```
g++ -Wall -o "%e" "%f" -lm -lGL -lGLU -lglut
```



Note that Geany uses `"%f"` as a placeholder for the current file name and `"%e"` for the filename without the file extension.

A Brief Explanation of the Compiler Flags Used

Single Source Files

A basic compile command might look like this:

```
g++ program.cpp
```

This will compile `program.cpp` into an executable file with the default name `a.out`. We can run the program like this:

```
./a.out
```

We can change the output name using the `-o` option:

```
g++ -o program program.cpp
```

Default C++ libraries are included by g++ by default. In order to include other libraries we need to use the `-l` (lower case L) option. Here we include the OpenGL (GL), OpenGL Utilities (GLU) and freeglut (glut) libraries:

```
g++ -o program program.cpp -lGL -lGLU -lglut
```

Multiple Source Files

With more complex programs that include multiple source files we need to compile all of the source files into object files. The object files are then linked together to create the final executable program. You can tell g++ to compile to an object file and stop before linking using the `-c` option. An example of compiling a program defined in `main.c` which includes code from `foo.c` and `bar.c`:

```
g++ -c -o foo.o foo.cpp
g++ -c -o bar.o bar.cpp
g++ -c -o main.o main.cpp
g++ -o program foo.o main.o bar.o
```

This is an efficient way to handle compiling a program from many source files. By storing the object files you can avoid recompiling code until it is modified. Unmodified code can just be linked from the existing version. Since we're working with very small projects that compile almost instantly we can skip storing individual object files and let g++ handle it for us. Here is the command to compile the previous example:

```
g++ -o program main.cpp foo.cpp bar.cpp
```