The GCC Compiler
Quick Reference Guide
for CS61B
edited by K. A. Yelick
based on a handout by P. N. Hilfinger

`Gcc` is a publicly-available optimizing compiler for C, C++ and other languages that runs under various implementations of the Unix operating system (including HPUX for HPs and Linux for PCs). Full documentation is available on-line under Emacs. (use `C-h i` and select the GCC menu option). There used to be a separate version of `gcc` called `g++` for compiling C++ program, and while the `g++` command still exists, it is really the same compiler as `gcc`.

Running the compiler.

To translate a C++ file, *file-name*`.cc`, into a corresponding object file, *file-name*.o, use the command:

`gcc -c` *compile-options file-name*`.cc`

To link one or more object files — *f1*`.o`, *f2*`.o`, ... — produced from C++ files into a single executable file called *f*, use the command:

`gcc -o` *f link-options f1*`.o` ... `-lg++` *other-libs*

You can bunch these two steps — compilation and linking — into one with the following command, where `both-options` is the list of compiler and linker options:

`gcc -o` *f both-options f1*`.cc` ... `-lg++` *other-libs*

After linking, your executable file becomes, in effect, a new Unix command. You can run it by simply typing its name. If you have named you your executable something common like `test` or `hello` that is already a Unix command, you must explicitly indicate the program in the current directory. If your program also takes arguments, the full command then becomes:

`./`*f arguments*

Libraries

A *library* is a collection of object files that has been grouped together into a single file and indexed. When the linking command encounters a library in its list of object files to link, it looks to see if preceding object files contained calls to function not yet defined that are in the library. When it finds such a function, it then linking the appropriate object file from the library.
Libraries are usually designated with an argument of the form `-l`*library-name*. In particular, `-lg++` denotes a library of standard C++ routines and `-lm` denotes a library of various mathematical routines. They must be listed *after* the object or source files that contain calls to their functions. (If you go about looking for the libraries themselves, they have file name that start with `lib` and ends with `.a`, for example, `libg++.a` is the standard C++ library.) Because the files are compiled before the library flags in the linker command have been processed, using a library often requires a corresponding "include" file in the C++ source program. For example, in a `.cc` file that contains calls to the math library one adds the command:

```
#include <math.h>
```

This informs the compiler that there will be calls to the math routines and allows it to check the argument types passed to those routines.

Options

The following compilation and link options will be particular interest to us.

`-c` (Compilation option)
: Compile only. Produces a `.o` file from each `.cc` file without doing any linking.

`-D`*name*`=`*value* (Compilation option)
: In the program being compiled, define *name* as if there were a line:

  `#define` *name value*

`-o`*file-name* (Usually used as a link option)
: Use *file-name* as the name of the output file produced by `gcc` (usually, this is an executable file).

`-l`*library-name* (Link option)
: Link in the specified library. (See previous section on libraries.)

`-g` (Compilation and link option)
: Put debugging information for `gdb` into the object or executable file. Should be specified for *both* compilation and linking.

`-MM` (Compilation option)
: Print the header (`.h`) files used by each source file in a format acceptable to `gmake`. (Standard headers will be omitted.) Don't produce a `.o` file or an executable.

`-pg` (Compilation and link option)
: Put profiling instructions used for `gprof` into the object or executable file. Should be specified for *both* compilation and linking. *Profiling* is the process of measuring how long various portions of your program take to execute. When you specify `-pg`, the resulting executable program, when run, will produce a file of statistics called `gmon.out` by default. A program called `gprof` will then produce a listing of those statistics.

`-Wall` (Compilation option)
: Produce warning message about a number of things that are legal but dubious. The compiler will produce the `.o` files even though there are warnings, but you should *always* specify this flag and treat each warning message as an error to be fixed.

`-fno-builtins` (Compiler option)
: There is a temporary glitch in our version of `gcc` which causes some of the above warnings to be suppressed. We will often use this flag to make sure we get all of the warnings.