

GNU Compiler Collection (gcc)

Humayun Kabir

Professor, CS, Vancouver Island University, BC, Canada

gcc

- gcc
 - stands for GNU Compiler Collection
 - a popular console-based compiler for UNIX platforms and others; can cross-compile code for various architectures
 - gcc to compile C programs; g++ for C++
 - can actually work with also ADA, Java, and a couple other languages
 - gcc performs all of these:
 - preprocessing,
 - compilation,
 - assembly, and
 - linking
 - we are to use it for our C++ labs and assignment
- As always: there is man gcc

gcc Options

- There are zillions of them, but there are some the most often used ones:
 - To compile: `-c`
 - Specify output filename: `-o <filename>`
 - Include debugging symbols: `-g`
 - GDB friendly output: `-ggdb`
 - Show all (most) warnings: `-Wall`
 - Be stubborn about standards: `-ansi` and `-pedantic`
 - Optimizations: `-O`, `-O*`

gcc Options: -c

- gcc performs, preprocessing, compilation and assembly of the source file without linking.
- The output are usually object code files, .o; they can later be linked and form the desired executables.
- Generates one object file per source file keeping the same prefix (before .) of the filename.

gcc Options: -o <filename>

- Places resulting file into the filename specified instead of the default one.
- Can be used with any generated files (object, executables, assembly, etc.)
- If you have the file called source.c; the defaults are:
 - source.o if -c was specified
 - a.out if executable
- These can be overridden with the -o option.

gcc Options: -g

- Includes debugging info in the generated object code. This info can later be used in gdb.
- gcc allows to use -g with the optimization turned on (-O) in case there is a need to debug or trace the optimized code.

gcc Options: -ggdb

- In addition to -g produces the most GDB-friendly output if enabled.

gcc Options: -Wall

- Shows most of the warnings related to possibly incorrect code.
- -Wall is a combination of a large common set of the -W options together. These typically include:
 - unused variables
 - possibly uninitialized variables when in use for the first time
 - defaulting return types
 - missing braces and parentheses in certain context that make it ambiguous
 - etc.
- Always a recommended option to save you from some “hidden” bugs.
- Try always using it and avoid having those warnings.

gcc Options: -O

- Various levels of optimization of the code
- -O1 to -O3 are various degrees of optimization targeted for speed
- If -O is added, then the code size is considered
- -O0 means “no optimization”
- -Os targets generated code size (forces not to use optimizations resulting in bigger code).

gcc Options: -I

- Tells gcc where to look for include files (.h/.hpp).
- Can be any number of these.
- Usually needed when including headers from various-depth directories in non-standard places without necessity specifying these directories with the .c files themselves, e.g.:

#include "myheader.h" vs.

#include "../foo/bar/myheader.h"

gcc Options: -L and -l

- **-L** tells gcc where (directory) to look for nonstandard library object files.
- gcc knows the path of standard library objects.
- **-l<library>** tells gcc to link with specific library. For example, if your code uses math library, you need to use
 - **gcc example.c -lm -o example**

gcc Example

- For example, if you have the following source files in some project of yours:
 - ccountln.h
 - ccountln.c
 - fileops.h
 - fileops.c
 - process.h
 - process.c
 - parser.h
 - parser.c
- You could compile every C file and then link the object files generated, or use a single command for the entire thing.
 - This becomes unfriendly when the number of files increases; hence, use Makefiles!
- NOTE: you don't NEED to compile .h files explicitly.

gcc Example

- One by one:
 - `gcc -g -Wall -c ccountln.c`
 - `gcc -g -Wall -c parser.c`
 - `gcc -g -Wall -c fileops.c`
 - `gcc -g -Wall -c process.c`
- This will give you four object files that you need to link and produce an executable:
 - `gcc ccountln.o parser.o fileops.o process.o -o ccountln`

gcc Example

- You can do this as well:
 - `gcc -g -Wall -ansi -pedantic ccountln.c parser.c fileops.c process.c -o ccountln`
- Instead of typing this all on a command line, again: use a Makefile.