Compiler Options | Coursera

coursera.org/learn/writing-running-fixing-code/supplement/8FNAo/compiler-options

Compiler Options

Compiler Options

Usually, you will want to name the resulting program something meaningful, rather than the default *a.out*. To change the output file name, use the *-o* option, and specify the output file name after it. For example, *gcc -o myProgram myProgram.c* would compile *myProgram.c* (as above), but instead of producing *a.out*, it would name the program *myProgram*.

Another option you will commonly want to use is *--std=gnu99*. This option specifies that the compiler should use the C99 standard with GNU extensions. There are a actually a few different versions of C (referred to as "standards" because they reflect a standardization of features). gnu99 will match what we describe in this course, and is generally a reasonable standard to program in.

Another useful pair of options are *-Wall* and *-Werror*. The first of these requests that the compiler issue *warnings* for a wide range of questionable behavior. As we discussed earlier, the compiler checks your program for certain kinds of errors. If it detects errors, it reports them to you and requires you to fix them before proceeding. Warnings are like errors in that the compiler will report a problem that it has detected, however, unlike errors, the compiler will continue and produce a program even if it warned you about something. The *-Werror* option tells the compiler to treat all warnings as errors—making it refuse to compile the program until the programmer fixes all the warnings.

Novice programmers often wonder why one would want to enable warnings to begin with, and especially to ask the compiler to convert warnings to errors. Errors prevent your program from compiling, so it seems like any way to avoid them and get to the compiled binary is a good thing. However, a programmer's goal is not just to produce any executable program, but one that works correctly at hand. Anytime the compiler can alert you to potential problems, you can fix them much more easily than if they lead to problems that you have to debug. A good analogy is purchasing a house. Think of the compiler as your inspector for the house. While you are eager to purchase and move into your new house, you would much rather have the inspector discover a potential problem before you buy the house and move in.

We strongly recommend compiling with at least the following warning options:

```
1

-Wall -Wsign-compare -Wwrite-strings -Wtype-limits -Werror
```

These options will help catch a lot of mistakes, and should not pose an undue burden on correctly written code.

Recent versions of gcc also support an option *-fsanitize=address* which will generate code that includes extra checking to help detect a variety of problems at runtime. Using this option is also strongly recommended during your development cycle. However, we will note that (at least with *gcc* 4.8.2 and *valgrind* 3.10.0) you cannot run a program built with this option in *valgrind*. The two tools detect different, but overlapping sets of problems, so use of both is a good idea—they just have to be used separately.