Using the Optimizer compiler flags

We'll cover the following

- Effects on debugging

There are a number of levels of “optimization” that you can request from the gcc compiler at compile time. You can read about them here. Turning on the optimizer flags asks the compiler to attempt to improve the performance (speed) of the code, typically at the expense of compilation time, (sometimes code size), and (usually) debugging ease.

If you type this:

```
gcc -Q --help=optimizers
```

You will get a long list of all of the various optimization options that are possible. Typically they are grouped into several “levels” of optimization which can be requested with a flag like -On where n is 1, 2, 3, etc.

See the gcc Optimize-Options for a full listing of what’s turned on when you request -O, -O1, -O2, -O3, etc.

Effects on debugging

As a general rule, in fact, you should not use optimization flags when you are still debugging your code with gdb. When your code is compiled with optimization flags, strange things can happen, for example, variables you have defined may no longer be present, some statements may be executed in different places than where they are coded (and sometimes not at all), etc.

Sometimes you can get impressive speedups with optimization, simply by compiling with an optimizer flag, you might get a speedup even of twice as fast.

As always, be sure to check out the links that follow if you want to gain a better understanding on the speed efficiency if a program.

We’re near the end of our bootcamp, and I’m very impressed if you’ve kept up till now. Awesome!

Stick around because we have many other utilities and libraries to discuss. They will do wonders for you in C.