Compiler Options

CSE 232 - Dr. Josh Nahum

Reading:

No reading, but the source for this lecture is:

https://gcc.gnu.org/onlinedocs/gcc/Invoking-GCC.html

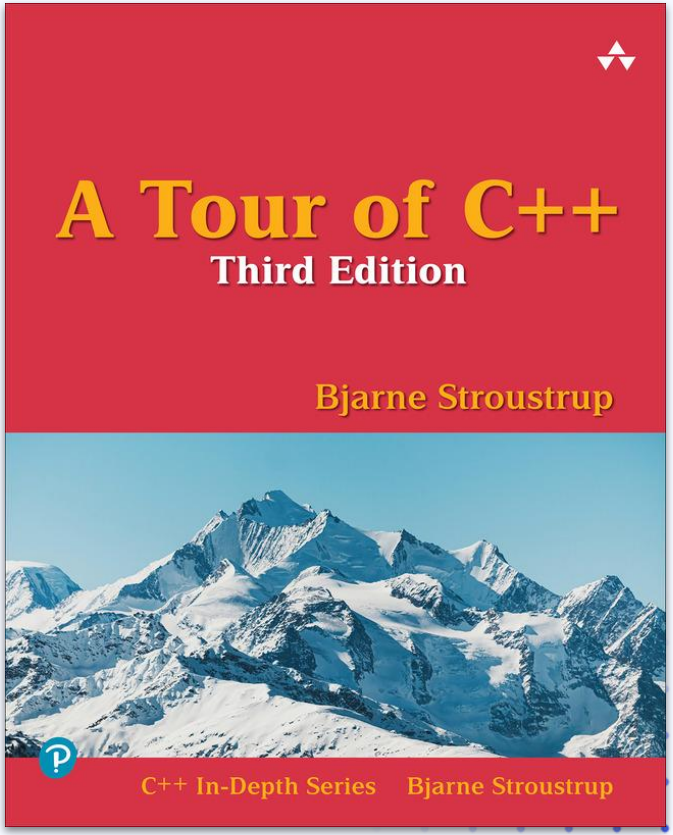

Table of contents

00

g++ and gcc

The compiler

g++

This is the program you were taught to use in the first week of CSE 232

gcc

This is the program that is commonly used to compile C language source code

Same

g++ is actually just **gcc** configured to compile C++ source code and link to the C++ library

01

Output and Dialect

Minimal Example

```
g++ main.cpp library.cpp
```

This terminal command instructs g++ (which is really just gcc) to compile two C++ source files, link them together into a single executable, and write that executable to a file (named by default "a.out").

Compile to object files

```
g++ -c main.cpp library.cpp
```

The **-c** option instructs g++ to compile each source file to an object file (a machine executable file, but they aren't linked into one program). In this case it creates the files: "main.o" and "library.o". It does almost all the compilation work, and it can be done **separately** (and independently) for each source file. This allows for faster compilation as only the altered source files need to be compiled.

```
g++ main.o library.o
```

This command links the object files into an executable ("a.out").

Output to named file

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

The **-o** option instructs g++ to output to the filename indicated next, instead of using a default name (like "a.out" or "main.o" as in the previous examples).

Dialect

```
g++ -std=c++20 main.cpp library.cpp
```

The **-std** option instructs g++ to use a particular version of the language, in this case C++20, when compiling and displaying warnings.

On my system, the default dialect (if **-std** isn't specified) is C++14, so be sure to include the language version!

02

Optimizations

Optimization Levels

-00 — Default optimization level. Optimized for short compilation time, and ensures that debugger and program match behavior.

-01 — Performs optimizations to reduce code size and execution time, but don't increase compilation time substantially.

-02 — Performs aggressive optimizations that will increase compilation time, but also increase the performance of the generated code.

-03 — A more extreme version of -O2

More Optimization Levels

`-Os` — Optimize for speed without increasing size (same as `-O2`, but without the optimizations that increase the size of the generated code)

`-Ofast` — Speed even if violates strict standard compliance. Mostly this means that some rules on how floating point math must behave are violated in the interest of performance.

`-Oz` — Performs aggressive optimizations for code size instead of speed. Code size will be minimized by generating slower code.

`-Og` — Very similar to `-O0`, but allows for `-O1` optimizations that don't interfer with debugging.

Example Usage

```
g++ -O3 main.cpp library.cpp
```

The program generated will be optimized to run as fast as possible (while still conforming to the standard's rules).

```
g++ -Oz main.cpp library.cpp
```

The program generated will be optimized to have the smallest possible code size (even though it may be slow). Useful on embedded devices with limited memory

Both of the above program are unlikely to work in a debugger and will likely be much slower to compile.

Disabling Assertions

```
g++ -DNDEBUG main.cpp library.cpp
```

The **-DNDEBUG** instructs the preprocessor (the first stage in compilation) to remove assertions from the generated code. This means the compiled code doesn't perform the assertion checks. Only enable this after you are sure the assertion checks are no longer needed for debugging.

Example Assertion:

```
#include <cassert>
int main() {
  int x{3};
  assert(x > 0);
}
```

04
Warnings and Debugging

Compiler Guidance

Warnings

Messages delivered by the compiler for issues that don't cause the compilation to fail are called **warnings**.
They usually indicate possible problems in your code and should always be examined.

Errors

Messages from the compiler that prevent it from compiling are called **errors**.

-Werror

```
g++ -Werror main.cpp library.cpp
```

The **-Werror** option instructs the compiler to treat all warnings as errors, meaning that any warning will cause compilation to cease.

This is the default option in the Codio Autograder to ensure that no student solution that generates warnings is allowed to compile

Warning Levels

`-Wpedantic` — Gives warnings for violations of the C++ Standard.

`-Wall` — Gives warnings for "questionable" code.

`-Wextra` — Exables extra warnings not already covered by `-Wall`.

Using Warnings

```
g++ -Wall -Wextra -Werror main.cpp library.cpp
```

```
int n{5};
if ((n > 1) == 2) ;
```

The if condition is always false. Using the **-Wall** option would raise a warning about that.

The if condition has an empty body. Using the **-Wextra** option would raise a warning about this fact.

The **-Werror** option causes the two previous warnings to cause compilation to fail.

Recommendations

Especially while you are learning C++, you should likely enable `-Wall`, `-Wextra`, and `-Werror` to help guide you away from common bugs.

Debugging

```
g++ -g main.cpp library.cpp
```

The **`-g`** option instructs g++ to include debugging information in the executable, so that debuggers (like gdb) are able to provide informative output about the state of the program as it runs.

Attribution

Please ask questions via Piazza

Dr. Joshua Nahum
www.nahum.us
EB 3504

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**