COLORADOSCHOOLOFMINES

TOPIC 5: COMPILATION

g++ ; -o, -c, -I (dash i), -L, -l (dash l), -Wall, -g

g++

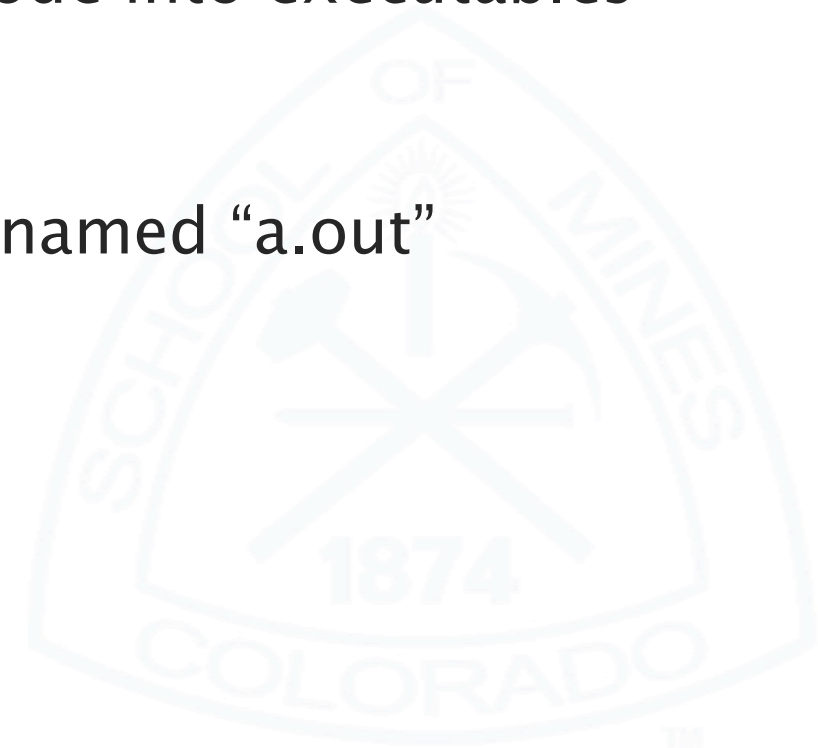
- g++ compiles C++ source code into executables

```
UNIX> g++ Hello_World.cpp
```

- By default, the executable is named “a.out”

```
UNIX> ./a.out
Hello World!
```

- What does “./” do?



g++ -o

- -o command line option names the executable

```
UNIX> g++ Hello_World.cpp -o Hello_World
UNIX> ./Hello_World
Hello World!
```

- Be very careful with using -o!
- What's wrong with the following command?

```
UNIX> g++ Hello_World.cpp -o Hello_World.cpp
```

g++ -c

- -c command line option compiles the source code into and an ***object file*** (.o file)
- Object files can then be ***linked*** to form an executable

```
UNIX> g++ -c Main.cpp

UNIX> g++ -c file2.cpp

UNIX> g++ Main.o file2.o

UNIX> ./a.out
```

g++ -c continued

- For example, given:
  - Main.cpp : contains main() that calls function1()
  - Function1.cpp : implements function1()
  - Appropriate header file (i.e., both files contain #include “header.h”, which defines function1() prototype)

UNIX> g++ -c Main.cpp

UNIX> g++ -c Function1.cpp

UNIX> g++ Main.o Function1.o

UNIX> ./a.out

Function 1 called!

g++ -I (dash i)

- -I (dash i) option tells the compiler where to find an external header (.h) file

UNIX> g++ file.cpp -I/path/to/header/file

- Note
  - The lack of whitespace between -I and the path
  - The header file name is NOT included in the path
  - The path can be absolute or relative

g++ -I (dash i) continued

- For example, given:
  - Main.cpp : contains main() and #include “header.h”
  - header.h : located in directory /example/path

```
UNIX> g++ Main.cpp -I/example/path
UNIX> ./a.out
Program ran successfully
UNIX>
```

g++ -L , -l (dash lowercase L)

- A library is a static (or dynamic) collection of object files
  - Libraries have .a extension
- -L option tells the compiler where to find an external library
- -l (dash lowercase L) tells the compiler the name of the library

UNIX> g++ file.cpp -L/path/to/library/file -lName_of_library

- Note
  - The lack of whitespace between -L and the path; -l and library name
  - The library file name is NOT included in the path for -L
  - The path for -L can be absolute or relative
  - The library name (for -l) does not contain .a or "lib"

g++ -L continued

- For example, given:
  - Main.cpp : contains main() and uses functionality from library libFoo.a
  - libFoo.a : A library file containing functions used in main, located in /Path/To/Library

UNIX> g++ Main.cpp -L/Path/To/Library -lFoo

g++ -I -L -l (dash i, dash L, dash lowercase L)

- We usually combine -I, -L, and -l
- For example, given:
  - Main.cpp : contains main(), #include "header.h", and function f1()
  - header.h : located in /path/headers; contains prototype for f1()
  - libFooBar.a : located in /path/libs; contains f1() implementation

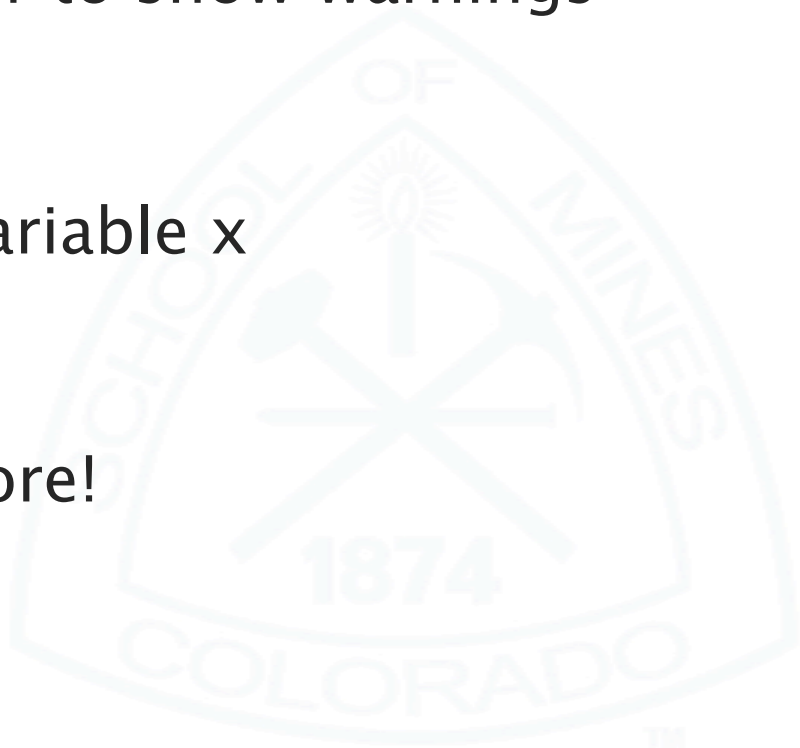
```
UNIX> g++ Main.cpp -I/path/headers -L/path/libs
  -lFooBar
```

- Why would developers do this?

g++ -Wall

- -Wall option tells the compiler to show warnings

```
UNIX> g++ Main.cpp -Wall
Main.cpp: WARNING: unused variable x
UNIX> ./a.out
Program is running just as before!
UNIX>
```



g++ -g

- -g option allows for debugging tools like gdb

UNIX> g++ Main.cpp -g

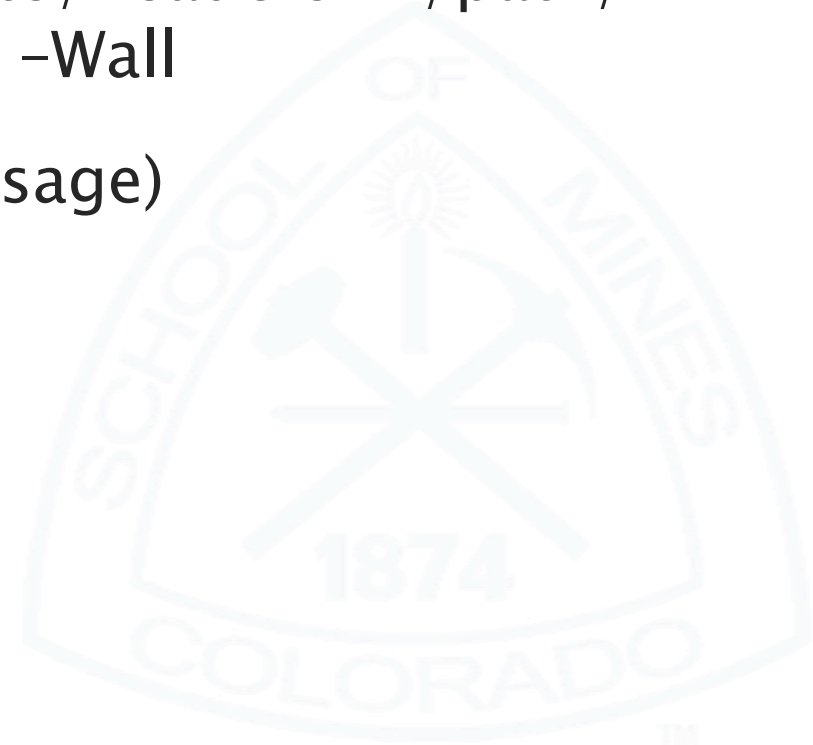
UNIX> ./a.out

Runs the same as before

- is g++ -g Main.cpp ok?
- Command line debuggers (e.g., gdb) are beyond the scope of this course!

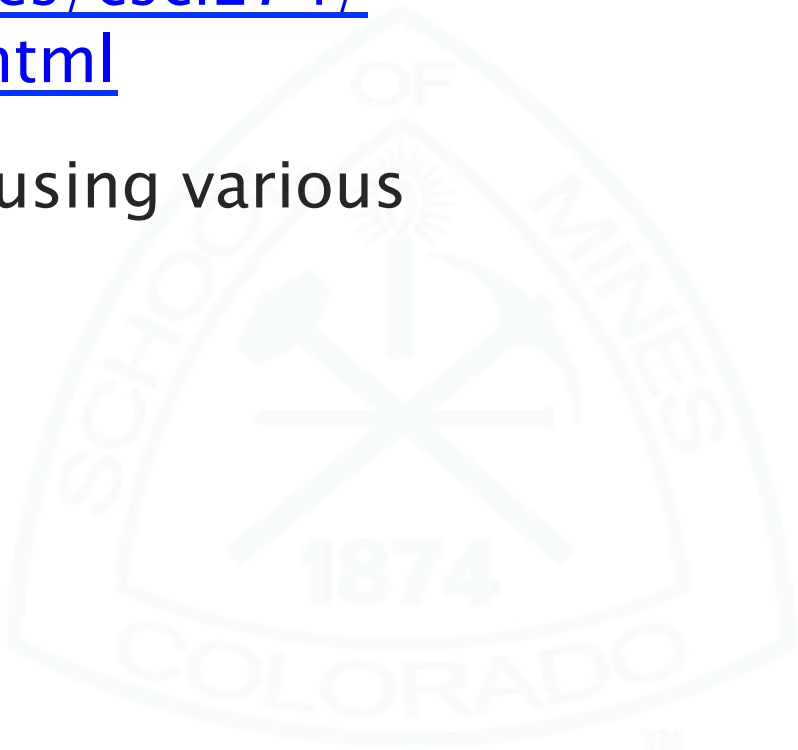
Combining it all

```
UNIX> g++ Main.cpp -I/path/to/headers -L/path/
  to/libs -lLibName -o Main -g -Wall

WARNING: (some Warning message)

UNIX> ./Main

Whoa!

UNIX>
```



Assignment 5

- http://eecs.mines.edu/Courses/csci274/Assignments/5_compilation.html
- Practice compiling programs using various command line options



The truth of XKCD