Makefiles
What are they for?

- From [https://www.gnu.org/software/make/](https://www.gnu.org/software/make/):
  - GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files.
  - In other words, it's a way to specify and automate your build process.
How is a C++ program compiled?

- When you type:
  - `g++ library.cpp main.cpp -o main`
  - ... what exactly is the compiler doing?
Appearance

main.cpp

library.cpp

Main (executable)
Reality

main.cpp

main.o (object file)

library.cpp

library.o (object file)

Main (executable)
Reality

Compile

main.cpp

main.o
(object file)

library.cpp

library.o
(object file)

Link

Main (executable)
The “-c” flag

- Turns out we can ask the compiler to only do the “compile” step:
  - g++ main.cpp -c -o main.o
  - g++ library.cpp -c -o library.o
- “-c” stands for compile only.
- The above commands creates “main.o” and “library.o”.
- It does not link them together.
Linking them together

- Pass the object files to g++, as if they were cpp files:
  - g++ library.cpp main.cpp -o main
  - g++ library.o main.o -o main
Why would you do that?

- Simple: if you changed “library.cpp”, you do not have to recompile “main.cpp”!
Now we can explain Makefiles!

Think of makefiles as recipes:

- The ingredients for “main.o” is “main.cpp”.
- The ingredients for “library.o” is “library.cpp”.
- The ingredients for “main” is “main.o” and “library.o”.

- To make “main.o”, do “g++ main.cpp -c -o main.o”
- To make “library.o”, do “g++ library.cpp -c -o library.o”
- To make “main”, do “g++ main.o library.o -o main”
In Makefile language:

main.o: main.cpp
    g++ main.cpp -c -o main.o

library.o: library.cpp
    g++ library.cpp -c -o library.o

main: main.o library.o
    g++ main.o library.o -o main
More compiler flags

- **-Wall**: Tells the compiler to warn you of possible errors in your code.
- **-O1, -O2, ...**: Tells the compiler to magically optimize your code.
  - Don’t do this if you are debugging your code!
- **-g**: (You’ve seen this last time) Tells the compiler to generate useful information for the debugger.
Now with make with more flags

main.o: main.cpp
    g++ -Wall -O2 main.cpp -c -o main.o

library.o: library.cpp
    g++ -Wall -O2 library.cpp -c -o library.o

main: main.o library.o
    g++ -Wall -O2 main.o library.o -o main
Now with make with more flags

main.o: main.cpp
    g++ -Wall -O2 main.cpp -c -o main.o

library.o: library.cpp
    g++ -Wall -O2 library.cpp -c -o library.o

main: main.o library.o
    g++ -Wall -O2 main.o library.o -o main
Automatic variables

- “$<” stands for “first item in the dependency list”

```bash
$<
```

```bash
cxx = g++ -Wall -O2

main.o: main.cpp
   $(cxx) $< -c -o main.o

library.o: library.cpp
   $(cxx) $< -c -o library.o

main: main.o library.o
   $(cxx) main.o library.o -o main
```
Automatic variables

- "$<" stands for "first item in the dependency list"
- "$@" stands for "the name of the target"

cxx = g++ -Wall -02

main.o: main.cpp
  $(cxx) $< -c -o @$

library.o: library.cpp
  $(cxx) $< -c -o @$

main: main.o library.o
  $(cxx) main.o library.o -o @$