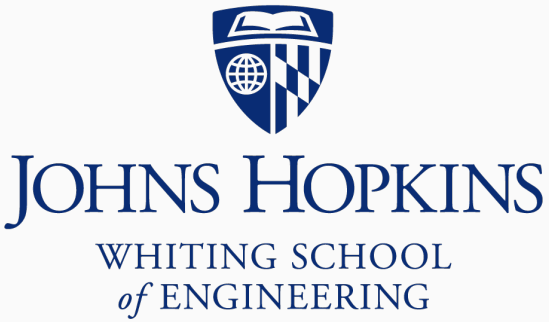
Program structure

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

Program structure

Big C projects are split across many `.c` source files

Header (`.h`) are part of the "glue" for combining `.c` files

Program structure

main.c

```c
#include <stdio.h>

int main() {
    float total = compound(100.0, 0.10, 10);
    printf("%0.2f\n", total);
    return 0;
}
```

interest.c

```c
#include <math.h>

float compound(float p, float r, int n) {
   return p * pow(1 + r/n, n);
}
```

`main` calls compound; compound is defined in a different `.c` file

Two problems:

- `main` needs to be preceded by a prototype for compound
- `main.c` and `interest.c` need to be compiled *together* somehow

Headers

main.c

```c
#include <stdio.h>
#include "interest.h"

int main() {
    float total = compound(100.0, 0.10, 10);
    printf("%0.2f\n", total);
    return 0;
}
```

interest.c

```c
#include <math.h>

float compound(float p, float r, int n) {
   return p * pow(1 + r/n, n);
}
```

interest.h

```c
float compound(float, float, int);
```

Header (`.h`) files contain prototypes & declarations allowing code in one `.c` file to use functions & variables in another

`interest.h` has a prototype for compound

- `#include "interest.h"` allows us to use it

Headers

When `#include`'ing a `.h` *you created*, use `" "` instead of `< >`

```
#include <stdio.h>    // provided by C
#include <assert.h>   // provided by C
#include "interest.h" // I wrote this
```

Headers

Compile the two `.c` files together by simply specifying both as arguments to gcc:

```
$ gcc main.c interest.c -std=c99 -pedantic -Wall -Wextra -lm
$ ./a.out
110.46
```

You don't have to tell gcc about the `.h` files; the `#include` statements take care of that

Headers

What if we forget to `#include "interest.h"`?

```
$ gcc main_noinc.c interest.c -std=c99 -pedantic -Wall -Wextra -lm
main_noinc.c: In function 'main':
main_noinc.c:4:19: warning: implicit declaration of function 'compound'
[-Wimplicit-function-declaration]
     float total = compound(100.0, 0.10, 10);
                   ^~~~~~~~
$ ./a.out
0.00
```

A compiler warning and a wrong answer; what happened?

- Hint: look back at notes on functions

Steps of compilation

We discussed compilation steps when we first saw "Hello, World!"

1. Preprocessor
    - Gather relevant source code
    - Handle `#include`'s and `#define`'s
2. Compiler
    - Gather preproessed code and compile to object code
    - If using `-c`, stop here and output `.o` files
3. Linker
    - Gather *object code* into a single executable file

Steps of compilation

Steps 1/2 and 3 can be separate

```
# Steps 1 & 2 -- preprocess and compile to separate .o's
gcc main.c -c -std=c99 -pedantic -Wall -Wextra -lm
gcc interest.c -c -std=c99 -pedantic -Wall -Wextra -lm

# Step 3 -- combine .o's into single executable
gcc -o main main.o interest.o
```

Or they can all happen at once

```
# Steps 1, 2 & 3
gcc -o main main.c interest.c -c -std=c99 -pedantic -Wall -Wextra -lm
```