Recitation 1

TA Hanxiong Chen
hc691@rutgers.edu

Header File

- What is header file?
  - A header ***file*** is a ***file*** with extension .h which contains C function declarations and macro definitions to be shared between several source ***files***.
- Is it necessary?
  - Not really.
- Why we need header file?
  - Easy to share your function or variables between several source files.
  - Get rid of redefine issues (IF YOU USE HEADER FILE “CORRECTLY”)
- What is in the header file?
  - Declaration of variables or functions
  - E.g. void func(int a); extern int i;
  - ATTENTION: You can ignore the “extern” when you declare a function, but you cannot ignore it when declare a variable. Don’t initialize the variable, or it will be defined.

Header File (cont’d)

- You may have seen other’s header file like this ->
- You always use #include at the beginning of your source code. What does that mean?

```
#ifndef _A_H_

#define _A_H_

void func(int a);

extern int v;

#endif
```

Header File (cont'd)

- What does "#include" do?
  - It just copy things!

Example (include .c):

```
File.c:   int a = 0;
          void func(int a){};

Main.c: #include "File.c"          ---->
          int b = 0;
          int main()
          {
             func();
             return 0;
          }
```

During preprocessing stage

```
int main.c:
int a = 0;
void func (int a){};
int b = 0;
int main()
{
   func();
   return 0;
}
```

Example (include .h):

```
File.h:   extern int a;
          void func(int a); // no brace
File.c:   int a = 0;
          void func(int a){};

Main.c: #include "File.h"          ---->
          int b = 0;
          int main()
          {
             func();
             return 0;
          }
```

During preprocessing stage

```
int main.c:

extern int a;
void func (int a);
int b = 0;
int main()
{
   func();
   return 0;
}
```

Header File (cont'd)

- What may go wrong if you "include" something?
  - Redefine!!
  - Eg. "main.c" includes "liba.h" and "libb.h" ; "liba.h" includes "libb.h" ← BAD!!
- How to solve the problem?
  - #ifndef _XX_H_
  - #define _XX_H_
  - Your declaration;
  - #endif

Use Macro to solve this problem!

If you are using Visual Studio, you can also use #pragma once (Don't use this if you want your code to be compiled on different platforms)

Header File (cont'd)

Bad Ideas:

- Don't define global variables in a header file!
  - What's wrong?
  - Something wrong when link multiple object files. Macro cannot help!!
  - Macro can only work within one source file.
- Don't use "static" in header file. (unless you are very profession in C).
  - If you use static to define a variable in .h file, you will not have redefine issues.
  - You cannot treat this variable as a global var, because it's static! It's localized!

Header File (cont’d)

Some figures to explain the issues.

Please see the figure I draw on the board!

Header File (cont'd)

Tips on writing header file:

1. Write your definitions of var and func in .c file
2. Write your declarations of var and func in corresponding .h file
3. Write "type definition" and "macro definition" in .h file.
   a. typedef struct t{}t;
   b. #define PI 3.14 //The value of PI cannot be changed in other places
4. Add #ifndef #define #endif in your .h file.
5. Recommend to include the xx.h in its own xx.c file. (not necessary)

Try to write your code to test what I said!

Practice makes perfect!!!

Macro

- How to define macro?
  - It’s easy!
  - Eg. #define NAME value
  - Please use capital characters as the name
- How does it work?
  - The macro name will be simply replaced by the value during the preprocessing
- When to use it?
  - Reduce typing work
  - **Want to change the same value at multiple places**

Question

0. What's wrong with this #define line?

```
#define N 10;
```

1. Suppose you defined the macro

```
#define SIX 2 * 3
```

Then, suppose you used it in another expression:

```
int x = 12 / SIX;
```

What value would x be set to?

Question (cont.)

Take a char, inspect its int value and return its corresponding int value. e.g.

```
int test = my_atoi('5');
if (test == 5)
{
        return 0;
}
Else
{
        return -1;
}
```

Take a string of any length, scan its chars until you hit the '\0' and return the entire string's int value. E.g.

```
int test = my_atoi("512");
if (test == 512)
{
        return 0;
}
Else
{
        return -1;
}
```