C Header Files

How to use C Header Files to separate a program into multiple files

Published Feb 29, 2020

Simple programs can be put in a single file, but when your program grows larger, it's impossible to keep it all in just one file.

You can move parts of a program to a separate file, then you create a **header file**.

A header file looks like a normal C file, except it ends with `.h` instead of `.c`, and instead of the implementations of your functions and the other parts of a program, it holds the **declarations**.

You already used header files when you first used the `printf()` function, or other I/O function, and you had to type:

```
#include <stdio.h>
```

to use it.

`#include` is a preprocessor (https://flaviocopes.com/c-preprocessor/) directive.

The preprocessor goes and looks up the `stdio.h` file in the standard library, because you used brackets around it. To include your own header files, you'll use quotes, like this:

```
#include "myfile.h"
```

The above will look up `myfile.h` in the current folder.

You can also use a folder structure for libraries:

```
#include "myfolder/myfile.h"
```

Let's make an example. This program calculates the years since a given year:

```
#include <stdio.h>

int calculateAge(int year) {
  const int CURRENT_YEAR = 2020;
  return CURRENT_YEAR - year;
}

int main(void) {
  printf("%u", calculateAge(1983));
}
```

Suppose I want to move the `calculateAge` function to a separate file.

I create a `calculate_age.c` file:

```
int calculateAge(int year) {
  const int CURRENT_YEAR = 2020;
  return CURRENT_YEAR - year;
}
```

And a `calculate_age.h` file where I put the *function prototype*, which is same as the function in the `.c` file, except the body:

```
int calculateAge(int year);
```

Now in the main `.c` file we can go and remove the `calculateAge()` function definition, and we can import `calculate_age.h`, which will make the `calculateAge()` function available:

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

int main(void) {
  printf("%u", calculateAge(1983));
}
```

Don’t forget that to compile a program composed by multiple files, you need to list them all in the command line, like this:

```
gcc -o main main.c calculate_age.c
```

And with more complex setups, a Makefile is necessary to tell the compiler how to compile the program.

(https://swift.flaviocopes.com)

(https://htmlcss.flaviocopes.com)

(https://javascript.flaviocopes.com)

(https://nodejs.flaviocopes.com)

(https://nextjs.flaviocopes.com)

(https://react.flaviocopes.com)

(https://databases.flaviocopes.com)

(https://linux.flaviocopes.com)