Notes for chapter 4 of the C book

October 23, 2017

Introduction

In this chapter we have seen several ways to structure a program. There are functions to keep actions isolated, external variables in order to make an application have a state, scopes for avoiding of name conflicts and headers to make an interface between different files.

Function

They are called functions, but it's not the best name. They are actually procedures because they allow to execute some code based on input parameters, so they are not really functions. They can also have an output value and they can modify the state of the program, which makes it possible to return several values.

External Variables

This is something complicated and in order to undestand them it's necessary to know how compilers work. And it's not something I know yet, so I cannot fully undestand the external variables.

An easy approximation of their behaviour would be to say that they are something like global variables. They are *external* to any function, and, like functions are often declared in headers.

A special case of external variables are static variables. When they are declared outside of functions they are like normal varibales, except that they cannot be visible from another file. When they are declared inside a function, they are only visible inside that function.

Header Files

This is like a contract for `*.c` files to implement, or like the visible part of the code of the `*.c` files. They usually have some forward function declarations and structure declarations (the book says they also comntain `extern` variable declarations, but I don't master them 100

Initialization

External and static variables are 0 when the execution starts. I don't really understand it, because it seems strange that a variable has a value, that it is not really assigned in the code, but I think it's for the purposes of filling large arrays with 0 when the program loads... I think they could have found another way to do it, with some built-in macros or special keywords.

The C Preprocessor

Here comes the most difficult part to understand. What I understand is that before compilation, some commands, contained in the code are executed, and another version of the source program is generated. The book says, that it's like text substitution. But if it was really the case, then a variable could have been declared outside a file and then used... But compilers don't accept it.

So, no. It's not that complicated. I made a test.

File1: cf1.c

```
int x = 0;
int dothings() { return x + 1; }
```

File2: cf2.c

```
#include <stdio.h>
#include "cf1.c"
int main()
{
  printf("%d\n", x);
  int y = dothings();
  printf("%d\n", y);
}
```

Makefile:

```
all:
gcc cf2.c  -o cf
```

Output:

```
0
1
```

So that means, that the border between files is not a real border, contrary to what the book says. The scope of an external variable is not from the declaration till the end of file. It's from the declaration to infinity.

Conclusion

SO it was a nice chapter. We had a nice opportunity to test some interesting things about the C programming language and I hope that this way we become a little better at understanding not only C, but also programming in general.