Lecture 3: Introduction to C

COMP26020 Part 1 (C) Lecture Notes

Pierre Olivier

These notes summarise the important points mentioned in the lectures. They are supposed to be a help for revising and not a way to avoid attending the live lectures and watching the videos. In other words, live lectures and videos may include examinable content that is not present in these notes.

The slides for this lecture are available here:
https://olivierpierre.github.io/comp26020-lectures/03-c-introduction.

Videos and recordings of live sessions can be found on the video portal: https://video.manchester.ac.uk/lectures.

Here we do a quick introduction to the C Programming language. We also learn how to write, compile and execute a few simple C programs.

C: Origin and Popularity

C is a relatively old programming language, it was developed in the 70s by Dennis Ritchie. It is still today one of the most favoured programming language. It is for example consistently in the top-3 of the TIOBE index that measures the popularity of programming languages[1].

C: Characteristics

In terms of pros, C has a very simple and extremely popular syntax. Many newer languages such as Java or Go take much inspiration from the syntax of C. C is also a **low level language** that efficiently maps to machine code and integrates well with assembly. It gives a high degree of control over the hardware, for example contrary to many languages C lets the programmer write at an arbitrary virtual addresses in memory. Because of its simplicity C is also very fast, and also allows programs to have a low memory footprint. This all comes at a relatively high cost which is a very large area to make mistakes, resulting from the high degree of control over the hardware. The compiler will not catch every mistake. And some errors may lead to **undefined behaviour** that is sometimes hard to debug. Still, the benefits of may make that it is the default programming language in many domains, including systems software such as operating systems or virtual machine monitors, but also high performance computing, embedded systems, among others.

Popular Software Written in C

A lot of extremely popular software is written in C:

- The Linux kernel, the most popular OS in HPC, cloud/servers, embedded systems
- The Xen hypervisor, one of the most popular virtual machine monitors.
- The GNU Libc standard library, used by most programs in many Linux distributions to interface with the OS.
- Apache, NGINX, and Redis, that are among the most popular web / key-value store servers.
- Git, which is by far the most popular version control system.

Hello World in C

This C program simply writes `hello world` to the console, also named the standard output:

[1] https://www.tiobe.com/tiobe-index/

```c
#include <stdio.h>
int main() {
    printf("hello, world!\n");
    return 0;
}
```

Before studying it more in details we can build and run it on the Linux command line as follows:

```
$ gcc listing1.c -o listing1
$ ./listing1
hello, world!
```

The first line with `#include` allows us to get access to functions from external libraries. We need the `printf` function from `stdio.h` to be able to print to the standard output.

Next we have a function named `main`. In C functions are implemented as follows. We first have the return type, here `int`, followed by the function name, `main`. Then between braces we have the function body, containing the function's code. In C `main` represents the program's entry point, i.e. the first function to run when the program is launched.

`main`'s body is composed of two statements. In C statements ends with a semicolon. The first statement is a function call: the `printf` function is called with a parameter between parentheses. `printf` is used to print something on the standard output, and that something is passed as a parameter. It is a character string, delimited by double quotes. `'\n'` is a special character defining a carriage return. Finally, the second statement uses the return keyword that returns from the current function to the calling context. We return the integer value zero, a common return code indicating successful execution. Because we return from `main`, this actually triggers the exit of the program with return code 0.

Compilation

The compilation is the process of creating an executable program from the source code. For this we use a tool named the **compiler**. The C compiler we will use is named the GNU C compiler, GCC.

A compiler takes the source code as input, here it's `listing1.c`. It generates the machine code ready for execution on the CPU and puts it into an executable, here `listing1`. `.c` is a common file name extension for C source code. Executables generally don't have a file name extension on Linux.

Concretely, we call the compiler as follows on the command line: `gcc`, followed by the name of the source file `listing1.c`, followed by `-o` to indicate the name of the output program, `listing1`:

```
$ gcc listing1.c -o listing1
```

Compilation: Warnings and Errors

The compiler will check and report some programming mistakes. There are two levels of issues reported by the compiler:

- An **error** is unrecoverable, and the compilation process will stop upon detecting it
- On the other hand, **warnings** may or may not be indicative of an issue and the compilation continues.

Compilers are very advanced today and warnings are almost always indicative of an issue: they should always be fixed. An important thing to note is that when multiple issues are listed, it is important to fix them in the order they are reported, because mistakes down the list may result from the ones up the list.

Comments

Comments are annotations in the source file that will not be processed by the compiler. It is important for a programmer to comment code a bit to explain what it does: for other people to understand his/her code, but also for himself/herself to dive back into complex pieces of code written a long time ago. There are two comments style supported in C:

```c
/* style 1 */
// style 2
```

Everything that is between the `/*` and `*/` delimiters is commented. One can also use the `//` to comment the rest of the line.