SYS2/CCCP: Crash Course on C Programming

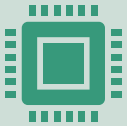
For Java and Python Programmers

Dr David Griffin

What's this course?

A crash course on C designed for people who have already programmed in Java and Python

Introduces the C language, and the bits that make it different to other languages



Not an introduction for programming

This course assumes you know how to program in Java (and a little bit of Python)

Course Structure

Online Mini-lectures

Introduce what you need to know for SYS2

Each lecture has its own “assignment”

Practical Sessions

Put things into practice

Do the assignments from each lecture

What is C?

- C is probably the oldest (1972) programming language still in widespread use
- Used in
  - Legacy Systems
  - Device Drivers
  - Low-level code
  - Implementation of other programming languages, especially interpreted languages (Python, Perl, Ruby, JavaScript, PHP, SQL databases…)

Why use C instead of Java or Python?

Biggest reason: Because you must

- Lots of big systems already exist in C – impractical to port them to another language (e.g. Windows, Linux, MacOS)

Predictable performance

- C is not a garbage collected language, so its performance is much more predictable than Java or Python
- C is (generally) a highly performant language

Need to be close to the metal

- C is about as close to the processor as you can get without using assembly code
- Lets you easily do lots of… esoteric things which device drivers need

Why is this course C for Java and Python programmers?

- Because these are the languages are taught to Computer Science students at York in the first year!
- Java and Python are both in the category *C-like* languages
- Both Java and Python took ideas from C when they were designed, and so there's a lot of similarities if you look for them
- However: Remember that C was designed about 50 years ago.
  - The core of the language dates from a time when computer memory was in kB, and monstrosities like COBOL and FORTRAN ruled the programming world
  - C introduced a lot of what makes modern programming languages nice…
  - And modern extensions to C have introduced some nice features
  - But it still lacks a lot of the things that *actually* make programming nice

Hello World in C

```c
#include <stdio.h>


int main() {

  printf("Hello, World!");

  return 0;

}
```

How Hello World in C Works

```
#include <stdio.h>

int main() {
  printf("Hello, World!");

  return 0;
}
```

Includes a file from the standard library which contains `printf`

Tells C where to start the program

Execute the `printf` function to print to screen

All C programs return an integer:
Zero = everything went well,
Nonzero = something went bad

Syntax in Hello World in C

Lines starting with # are (for now) magic that change into something else before compiling the program

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

These <> are a different type of magic to indicate “Standard Library”

```
int main() {

    printf("Hello, World!");


    return 0;

}
```

Curly braces and semicolons work like they do in Java

You should use whitespace to make code readable, but it’s not Python. The { } are used to indicate a block of code, not indentation.

Not so bad, right?

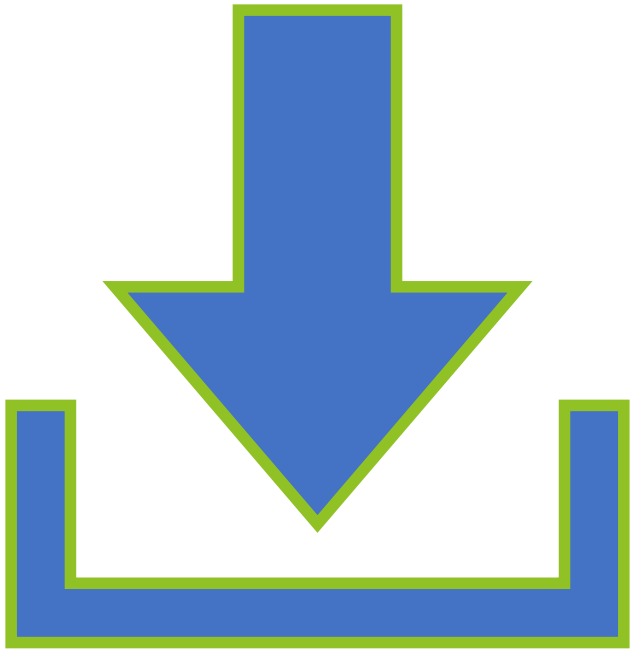
Simple programs like Hello World are easy enough, but C has a lot of features which make things more complicated

This course will be focusing on those complicated features, and why you would want to use them

We'll be covering:

Data Types and Structures
Program Flow, Pre-processing and Compilation
Operators: Normal/Bitwise/Prefix/Postfix
The C Memory Model, Pointers and References
Putting it altogether for a low-level program

Assignment Overview: hello_world.c



The assignment for this lecture is fairly simple: Compile and Run a hello_world.c file on the University computers

This will also give you a chance to get to learn how to use Linux if you haven't used it already

Normally the lecture would end here… but the next couple of slides have a few hints for compiling C programs

Using Linux on the University Machines

If the computer is running Windows, reboot

If it's off, turn it on (this should go without saying)

After the computer turns on, you will be presented with a menu. Use the arrow keys to select "Ubuntu", which is the Linux Distribution used at York

You can log in with your normal University username and password

Programs you'll be using

For this course, we assume you're using the standard Gnome desktop.

We'll be using a simple text editor to edit code. Click "activities" in the top left, and then type "text editor" to find it.

You'll also want to open a "Terminal" as well. Either find it in "activities" or press Ctrl+Shift+T to open one.

Using the Terminal

- A terminal gives you a command line interface. You can type in commands and run them by pressing Enter. Here are some useful commands and tricks:
  - `cd` – change directory
    - `cd Desktop` to go to your Desktop folder.
    - `cd ..` goes up one folder.
  - `ls` – list files and directories in the current directory
  - `rm` – Delete a file (use `rmdir` for an empty directory)
  - `man` – Open a manual page. Press `q` to quit.
    - `man ls` will give you more info on the ls command
  - Tab – use the tab key for autocomplete
  - Up/Down – use the up and down arrow keys to go through commands you've run before; very useful to recompile something

Using man

- The manual is very powerful and has multiple sections.
  - By default it goes to Section 1: Executable Programs and Shell Commands
- For C programming, Section 3 gives you information on functions in libraries
  - `man 3 printf  # Look up information on the printf function`
- Provided that the code has been documented properly and the manual updated appropriately, this will work with all libraries installed on the system

Compiling/Running hello_world.c

- Download the hello world code from the VLE
- Use cd to switch into wherever you saved the code
- Type `gcc hello_world.c -o hello_world` to compile the hello_world.c file
  - `-o` is an argument to gcc, the C compiler we're using. We'll look at these in more detail later
  - You can use Tab to autocomplete the name of the hello_world.c file
- Type `./hello_world` to run your compiled program
  - `./` is needed to run executables that are in the current directory. Without it, Linux will look for programs in something called the path, which by default is things that are installed like ls or gcc.