Exercise 1: writing and compiling a C program
===============================

You should do all this exercise as a *normal* (non-root) user. You are going to type in, compile and run a 'C' program from scratch.

Create the source file

Using your favourite editor (vi, ee, joe or whatever), create a file called hello.c containing the following text:

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

This is not a programming course, so you don't need to understand everything which is there. What we are really concerned with is how to *build* the program, rather than how to write it. But in summary:

* You are pulling in definitions of standard input/output functions (stdio)
* "main" is the entry point of your program; "void" means it takes no arguments
* printf(...) writes a screen to standard output. \n means newline.
* return 0 is the exit code from your program
* note that each C statement is terminated with a semicolon: ;

Compile and run

Once you have saved this file, you need to compile it like this:

```
    $ gcc -Wall -o hello hello.c
```

If you get a shell prompt back, it was successful. If you get an error message, it will include a line number; the mistake is probably somewhere around that line. Go back into the editor and correct the program. Extra spaces are not important, but a missing bracket or semicolon will make the file invalid.

If everything was OK, then you can run the program:

```
    $ ./hello
    Hello, world!
    $
```

Ask an instructor if you see anything else.

The reason you need to put ./ in front of the program name, is that the shell only looks for programs in the PATH (and your home directory is not in the PATH). Hence you need to tell the shell explicitly to look for the file in the current directory, which is "."

Extras (if time available)
----

* Have a look at the generated binary. How large is it? What does it actually look like? Useful commands are:

```
        $ ls -l hello
        $ less hello
        $ hexdump -C hello | less
        $ strings hello | less
        $ file hello
```

  (Although your program contained only two instructions, there is quite a lot of startup code which the compiler has to include)

* Check the exit code from the program. It's in the special shell variable "$?", so you can see it like this:

```
        $ ./hello
        $ echo $?
```

* Modify the program (e.g. to print a different message or return a different exit code). Rebuild it and re-run it.

* Add comments to your C source file. Comments are formatted like this:

```
    /* this is a comment */
```

  Comments make your source code more readable, but are ignored by the compiler and do not affect its output.