Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming (2021/22)
Laboratory Session Week_1 – Introduction

Learning Objectives

- To become familiar with the Linux programming environment.
- To introduce the main software development tools that will be used in this course.
- To compile and run simple programs written in the C language.

1. Linux environment

Login/logout and Desktop

You should login to one of the computers in the ITL Computing Lab at the start of the session, regardless of whether you are attending in person or remotely; amongst other things, you will need your EECS username and password (not your QMUL computer account). At the end of the scheduled laboratory session, please remember to logout from the (remote) computer you used.

If attending the lab session in person, you can directly login to one of the computers in the ITL Computing Lab. Otherwise, remote login to a "Linux" computer by following the instructions at http://support.eecs.qmul.ac.uk/services/rgate/; then choose the highlighted option in **Figure 1**.

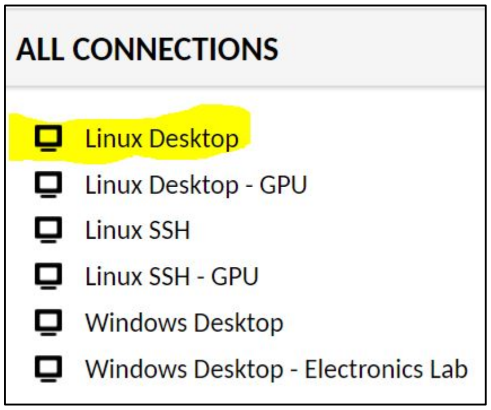

Figure 1

If you have a Windows OS computer and would prefer to install the necessary software on it to allow you to compile and run C code, please refer to the **Appendix – Installing Cygwin: Brief Tutorial** at the end of this document. Cygwin is “a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows”[1].

Note: It is not possible to provide technical support for installing Cygwin on your local computer.

[1] Quote taken from https://www.cygwin.com/.

User Shell

You will work from the user shell, in which you need to type in commands on a line. Here are some examples of commands that you will find useful throughout this course:

Command[2]	**Brief Description**
`ls -l <dirName>`	List information about files (in directory **`dirName`**, if specified)
`cd dirName`	Change to directory **`dirName`**; move to parent directory with **`'..'`**
`mkdir dirName`	Create a directory (i.e. a collection of files) called **`dirName`**, if it does not already exist
`rm fileName`	Remove (i.e. delete) the file called **`fileName`**[3]
`rmdir dirName`	Remove (i.e. delete) the directory called **`dirName`**[2]
`cp <f> <dir>`	Copy file **`f`** to directory **`dir`**
`cat <fileName>`	Display the contents of file **`fileName`** on the standard output stream (i.e. usually the screen)
`mv <f1> <f2>`	Rename file **`f1`** to file called **`f2`**
(upper arrow)	Repeat previous command; this is extremely useful!
`exit` (*or* `logout`)	To exit from the command line window; you can also exit by clicking the "X" button on the top right corner of the shell window.

Table 1 – Useful shell commands

Notes:

- For each of the commands in **Table 1**, you can obtain more information about them by typing in, on the command line, the name of the command followed by **`--help`**. For example, if you want to know more about how to use the **`ls`** command, type:

```
$ ls --help
```

Figure 2

- For a comprehensive list of shell commands (as well as how to use them), see e.g. http://linuxcommand.org/lc3_learning_the_shell.php.

TRY THIS

Practise using the various commands listed in **Table 1**, so you can easily manage the files and directories that you will be creating for this course.

At this point, it will also be useful to create a directory called e.g. **`ECS501U_LabWork`** (perhaps with several sub-folders, one per week), to store the lab work that you will be doing in the coming weeks.

[2] All commands are given here in their simplified form only; commands may have other options that may be (at times) useful.

[3] Be careful when using this command; there is no 'un-delete' option!

2. Text editors [Writing C source code]

To write and save your C programs (before you compile, run and debug them), you can use a variety of text editors, e.g. **nano**, **vi** and **Kate** (short for KDE Advanced Text Editor)[4].

Below are basic instructions to create and save a simple C program, such as the `helloWorld.c` example already covered in class (see **Figure 3**), using the GNU **nano** text editor. For information on how to use the other suggested text editors, **vi** and **Kate**, see:

- **vi** → Simply type `vi` (or `vim` – short for VI improved) on the command line; this will bring up a summary of the commands. More detailed information is, e.g. at http://ex-vi.sourceforge.net/.
- **Kate** → Simply type `kate` on the command line to start up the tool. More detailed information is, e.g. at http://kate-editor.org.

```
#include <stdio.h>          /* include the standard I/O library */

int main() {
  printf("Hello World\n"); /* print out the message to the screen */
  return 0;
}
```

Figure 3

GNU nano
To get started, type the following on the command line:

```
$ nano -c helloWorld.c
```

Figure 4

The flag `-c` tells **nano** to display the line number and position of your cursor at the bottom of the screen. The editor will know from the file extension that the program is C language source code, so it will highlight the text of your program in helpful colours.
Having successfully created the file, type in the code as shown in **Figure 3**; make sure you include all the brackets, curly braces and semicolons in the correct places.
After entering the program code into the editor, save the result by using `Ctrl-o` to write out the new text to the file and then `Ctrl-x` to exit the text editor. The next natural step would be to compile and run the program, which you will be doing soon …

For more information about the **nano** text editor and how to use it, you can:

- type `nano --help` on the command line;
- see, e.g. http://www.nano-editor.org.

TRY THIS
Practise using the various text editors (and at least **nano**) by creating and saving other C programs, e.g. the examples covered in class.

[4] It is up to you which text editor you use, but you may find that you may end up using all of them, depending on the specific task you want to achieve.

3. GCC GNU C compiler [Compiling and running C programs]

Now that you have created and saved some C programs, you need to compile and then attempt to run those programs. In this course, we will be using the GNU C compiler, via its **gcc** program on the command line.
We will be using **gcc** version 8.2.0[5] and you should be able to confirm this by typing the following on the command line (see **Figure 5**); you may need to first type the **`module load gcc/8.2.0`** command, to ensure that is indeed the **gcc** version running in the machine you are using:

```
$ gcc --version
```

Figure 5

Compiling and running your first C program
Open a terminal window if you do not have one already open, and make sure you are in the same directory where you saved your **`helloWorld.c`** program to[6].
If you are in the correct directory, invoke the **gcc** compiler by typing the following on the command line and hitting the return key:

```
$ gcc helloWorld.c
```

Figure 6

If you see any messages on the command line, this means that the compiler has detected some errors (or at least warnings) in the code that you saved in the file **`helloWorld.c`**; use the text editor to correct your C source code and then recompile the saved file[7].

If you copied the program from **Figure 3** correctly, then you should get no messages back from the compiler; however, now if you type **`ls`** on the command line, you should see that a new file called **`a.out`** has appeared in your directory. This is the executable program that the compiler created from your source code.
You can run the compiled program at the command prompt and see that the program works, by typing the following:

```
$ ./a.out
Hello World
```

Figure 7

By default, all programs you compile with just the command shown in **Figure 6** will generate an executable file called **`a.out`**; this is not very useful, as you may want (or need) to distinguish between different executables.
If you want to name your executable file (e.g. to be called **`exeFile`**) when you are compiling the source code (e.g. if **`sourceFile.c`** is the name of the source code file), type the following on the command line: **`gcc sourceFile.c –o exeFile`**

[5] This is the default package version installed on Linux desktops.

[6] You can check to see if your file exists by typing **`ls`** at the command prompt to list the contents of the directory – see **Table 1**.

[7] Later in the course, we will see how to debug compilation errors by using another tool: the GBD GNU Project Debugger.

Figure 8 shows what you would need to do if e.g., you want your 'helloWorld' program executable to be called **`helloWorld`**.

```
$ gcc helloWorld.c -o helloWorld
```

Figure 8

There are also several other options (or flags) that can be used with the **gcc** command; some of the most useful are listed in **Table 2**:

`gcc` Command options	**Brief Description**
`--help`	Get all information on the **`gcc`** command
`-o`	Name the program's executable
`-Wall`	Compile and display all warnings; very useful!
`-g`	Insert debugging code (i.e. **`gdb`**)[8]
`-l`	Compile with specified library[6]
`-E`	Generate pre-processor output only[6]

Table 2 – Useful `gcc` command options

Finally and as illustrated in class, you can also compile and run your C programs using the following command, shown for the example of the 'helloWorld' program (see **Figure 9**).
This command will only run your C program if it is able to compile the source code successfully (and in that case, saves it to file **`helloWorld`**).

```
$ gcc helloWorld.c -o helloWorld && ./helloWorld
Hello World
```

Figure 9

TRY THIS
Practise compiling and running other C programs, e.g. the examples covered in class.
You can also introduce errors in the source code and check what happens when you attempt to compile such code.

4. General exercises: program design and simple C functions

This section of the lab sheet contains exercises for you to practise what was learnt so far in teaching week 1, namely: basic C syntax (only simple variable declaration and the **`for`** loop) and a simple C function, i.e. **`printf()`**.

Please attempt the exercises by using only the C constructs that you learnt so far this week, and:
1. Write pseudo code to describe the required algorithm to solve the exercise (or draw up a flowchart), before writing and testing the actual code.
2. Add comments to your code.
3. Make your code neat, by using indentation and parenthesis (where appropriate).
4. Give meaningful names to functions and variables.

[8] This will be used later in the course.

Exercise 1
Write a program that displays a square shape filled with the character '*', with the side size as specified by a variable in the program. Save your program to a file called **`drawSquare.c`**.

Exercise 2
Write a program that displays a right angle triangle shape, filled with the character '*'. The two equal sides of the triangle are specified by a variable in the program, and the triangle's right angle should be displayed on the left hand side. Save your program to a file called **`rightAngleTriangle_left.c`**.

Exercise 3
Write a program that displays a right angle triangle shape, filled with the character '*'. The two equal sides of the triangle are specified by a variable in the program, and the triangle's right angle should be displayed on the right hand side. Save your program to a file called **`rightAngleTriangle_right.c`**.

Exercise 4 – **Homework**[9]
Use the result of exercises 1-3 to write a program that displays a parallelogram shape; e.g. if the equal sides of the triangles and the square side have a size of 4, then running the program would display the shape in **Figure 10**. Save your program to a file called **`drawParallelogram.c`**.

```
$ ./drawParallelogram
*
**
***
****
****
****
****
****
****
 ***
  **
   *
```

Figure 10

Hint: It will be useful to convert some of the previously written code into simple C functions.

ECS501U – END of Lab Week_1

[9] It is recommend that you only attempt to do this exercise, after the end of this week's ECS501U lectures.

APPENDIX – Installing Cygwin: Brief Tutorial

IMPORTANT:

1. This may be useful if you have a Windows OS computer and would prefer to install on your local machine, all the necessary software to compile and run C code.
2. It is not possible to provide technical support to install Cygwin in your local computer.

GENERAL STEPS to install Cygwin

1. Go to Cygwin home page: http://cygwin.com/
2. Download the setup file and run it.
   - Choose the set up according to your machine characteristics (i.e. 32-bit installation or 64-bit installation).
   - Try to choose a geographical close site for downloading, e.g. I downloaded from a Portuguese site (last dot is **pt**).

The current default installation of Cygwin does not include the C compiler **gcc**, therefore you need to select this package during installation. You can get all the installation information from the page http://cygwin.com/faq.html, which is briefly summarised below for the most important parts:

> "If you want to build programs, of course you'll need **`gcc`**, **`binutils`**, **`make`** and probably other packages from the "Devel" category. Text editors can be found under "Editors"."

The Cygwin setup is designed to make it easy to browse categories and select what you want to install or omit from those categories.

3. At the "Select Packages" screen (in the "Categories" view, at the line marked "All"), you see the word "default"; leave it as is because the full installation of Cygwin is greater than 800M bytes!
4. However, to get **`gcc`** you need to click on "Devel"; then all packages belonging to this category will appear, among which are[10]:
   - **`binutils`**: GNU assembler, linker, and similar utilities
   - **`gcc-core`**: GNU Compiler Collection (C, OpenMP)
   - **`gcc-g++`**: GNU Compiler Collection (C++)
   - **`gdb`**: GNU debugger
   - **`gitk`**: Git repository browser
   - **`make`**: GNU version of the 'make' utility

In the lines these packages are shown, you will see the word "Skip".

5. Click on it, and the latest version of the package will appear and therefore you will be selecting that package for installation.
   - Select those packages and then press the **NEXT** button to proceed the installation.
   - Sometimes selecting one item will automatic select others; do not modify this automatic selection.

The system should download the files and complete the installation. You should be able to open the Cygwin window, compile and run C programs.

[10] The full list of packages can be found at https://cygwin.com/packages/package_list.html.