operating systems lab - week 1: exercise

Nicolo Colombo

September 25, 2020

This lab is about getting started with the environment for this course and practising the basic concepts of C. Please be sure you have complete this exercise before attempting the Moodle quiz

Lab quiz1

Note that the quiz questions may require you to run, comment or modify some of the programs described below.

1 Getting started

1.1 Connect to the teaching server `linux.cim.rhuk.ac.uk`

Unix users Open the terminal and run

```
ssh yyyyxxx@linux.cim.rhul.ac.uk
```

where `yyyyxxx` is your college username, and enter your password to access the teaching server.

Windows users Launch the Windows SSH client `puTTY` [1], enter the following

```
linux.cim.rhul.ac.uk
```

in the empty field *Host Name (or IP address)* and click on *Open*. The client opens a new window where you are required to enter your college user name (normally four letters and three numbers) and password.

1.2 Create a new directory

You can now see the content and navigate in your home directory using the UNIX commands: `ls`, `cd`, `..` Create a new directory for the CS2850 labs by running

```
mkdir CS2850labs
```

move there and create a sub-directory called `week1`. We suggest you save and compile all programs you write for this exercise in this directory.

1.3 Write a C code

Choose one of the following command line editors

- emacs

[1] `puTTY` should be installed on all department's machines. If you work on your own Windows machine you can download it at download puTTY and install it as explained.

- nano
- vim

and create a new file called `helloWorld.c` by typing

```
editorName helloWolrd.c
```

Try to get familiar with the editor by copying into `helloWorld.c` the following C code

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

Once you are done, save and exit.

1.4 Compile your C code

Compile the C file `helloWorld.c` by running

```
clang -o helloWorld helloWorld.c
```

where `helloWorld` is the name you choose for the executable of `helloWorld.c`, and look at the content of directory `week1` with the `ls` command. You should find a new file, `helloWorld`, which is the executable of the C program `helloWorld.c`. What happens if you compile the same source file `helloWorld.c` without specifying the name of the output, i.e. if you remove the optional command input `-o helloWorld`?

1.4.1 Compile `helloWorld.c` using `gcc`

Redo the same by using the `gcc` compiler, i.e. compile `helloWorld.c` by running

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

Again, see what happens if you let the compiler choose the name of the produced binary file.

1.4.2 Compile Example 1 of Slides 1

What is the difference between the C code you have written into the file `helloWorld.c` and the code shown in Example 1 of Slides 1? Copy the code given in the slide into a new file, `helloWorld2.c` and try compile it with both `clang` and `gcc`. Observe and try to understand why both compilers produce a warning message. Compare the text of the warning message produced by the two compilers.

1.5 Run the executable

To run the binary file `helloWorld` created by the compiler, enter the following command (if you are in the same directory than the executable)

```
./helloWorld
```

Check that the program correctly prints the string `hello, world` on terminal. What happens if you run the executable of `helloWorld2.c`?

1.6 Debugging

The free system `valgrind` contains powerful debugging tools for Linux programs. The Valgrind's tool suite is already installed on `linux.cim.rhul.ac.uk`. We suggest you use it to automatically detect bugs in the programs you write for this course's labs. To see what may be wrong with your program, run the following

```
valgrind ./helloWorld
```

and have a look at the messages printed on the terminal. For the moment, this may look unnecessary and the messages you get quite trivial. But running such a sanity checks will become more and more important in the following weeks. One of the hardest part of learning C is to understand how to manage the memory allocated by a program and looking at the `valgrind` messages may save you hours of debugging work.

2 Control flow

The control structures `for` and `while` allows you to repeat an operation a given number of time. The usege and syntax are pretty similar to what you have seen for other languages. But feel free to check all the details on this C online manual.

2.1 Create loops with `for`

Save a new copy of `helloWorld.c` called `forHelloWorld.c` and modify it so that the program prints the string `hello, world` ten times. Use `Valgrind` to see if your program runs correctly and the heap usage of your program. Why is the heap usage of `helloWorld` and `forHelloWorld` the same? Remove the new-line operator `\n` in the argument of `printf` and observe what happens. Find a way to avoid the observed strange behaviour.

2.2 Create loops with `while`

Repeat what you have done in Section 2.1 by using a `while` statement.

2.3 Print iteration numbers

Have a look at this C online manual to understand how you can use `printf` for printing integer variables on the terminal. Create a new program `numberHelloWorld.c` that prints

```
i - hello, world
```

where `i`$= 1, ..., 10$, on ten different lines using the `for` statement. Modify the iteration rules in `for` so that the program:

i) prints `i - hello, world` for `i`$= 5, ..., 10$

ii) prints `i - hello, world` for $i = 1, 3, 5, 7, 9$

Note Your code should not include any `if` statements.

3 Sum of integers

Have a look at the C code shown in Examples 3 and 4 of the pdf document Slides 1 and write a new program called `sumOfIntegers.c` that i) compiles without warnings and ii) prints the following two-column table on the terminal

```
0       0
1       1
2       3
3       6
```

```
4       10
5       15
6       21
7       28
8       36
9       45
10      55
```

We suggest you follow the following steps:

1. examine the table above to understand how the second integer in each column is obtained from the first

2. start by declaring an `int` variable, `N`, that fixes the number of lines printed by the program. E.g. your program should produce the table above if `N` is declared as

   ```
   int N = 10;
   ```

3. use a `for` statement to iterate over the lines. Note that the first integer of each row can just be the iterator variable

4. change the iteration rules to print on screen only an arbitrary selection of lines. What is the problem with this solution? Run the program with different selection choices to understand what is going on.

5. introduce a conditional structure, `if(condition){}`, in the loop so that your program can be quickly set to print different row selections by changing the expression `condition`. For example, you may try to print

   i) only rows associated with even/odd iterations

   ii) only the last three rows

   iii) only rows was second entry is even/odd