An Introduction to HPC and Scientific Computing

CWM, Department of Engineering Science

University of Oxford

Wes Armour

Practical 3: Practical examples using the C programming language.

This practical will review some of the example codes presented in the lecture "Introduction to the C programming language". We will then use some of the material to build on these codes.

The learning outcomes of this practical are:

- To be familiar with basic C codes and be able to interpret them.
- To understand how variables are handled in a C code.
- To be familiar with operators and functions.
- To understand basic input and output.
- To have confidence to write your own basic C program.

All practicals for this course will be carried out on the Universities ARCUS-HTC computer. To understand how to use ARCUS-HTC see the introduction video and for more details watch the video demonstrations in lecture 3. As a reminder:

1. Connect to the university VPN (instructions can be found on canvas).
2. Open MobaXterm (Windows) or XQuartz (Mac) and open a ssh session.
3. Connect to "oscgate" by:
    a. Setting "*remote host*" to be oscgate.arc.ox.ac.uk
    b. Tick "*specify username*" and set it to teachingXY (where this is the username we have issued you with)
4. This will open a shell, from here you can connect to "arcus-htc" using ssh as follows:

    ssh –CX teachingXY@arcus-htc

Where teachingXY is the account that we have issued you with.

Instructions for this practical

Part A

1. If you have not done so clone the github repo for this CWM. To do this, at the command prompt type:

    $ module load git

$ git clone https://github.com/wesarmour/CWM-in-HPC-and-Scientific-Computing-2020.git

If you have already cloned the repo, pull it again to ensure you are working with the most up-to-date codes:

```
$ module load  git
$ git pull
```

2. Next you should navigate to the examples directory in the prac3 folder:

   $ cd CWM-in-HPC-and-Scientific-Computing-2020/practicals/prac3/examples/

3. Now use the Makefile provided to compile all of the source codes, simply type make:

```
$ make
gcc example_one.c -o example_one
gcc example_two.c -o example_two
gcc example_three.c -o example_three
gcc example_four.c -o example_four
gcc example_five.c -o example_five
gcc example_six.c -o example_six
gcc example_seven.c -o example_seven
gcc example_eight.c -o example_eight
gcc example_nine.c -o example_nine
gcc example_ten.c -o example_ten
gcc example_eleven.c -o example_eleven
```

4. Look through these codes and ensure that you understand them. Execute each one to try them:

   $ ./example_seven

   Circumference: 69.080002

Part B

It's now time to write your first C code. Imagine the following:

A farmer has had an unfortunate visitation from some naughty aliens. They've destroyed some of his crop:

He now needs to work out how much of his crop has been lost. To do this he's measured the radii (in meters) of all of the circles that have been left and stored these in a file called radii.txt:

$cd CWM-in-HPC-and-Scientific-Computing/practicals/prac3/code

He's also tried to write a C code to calculate how much barley has been lost. He's not really sure how to do this though so he's included comments and not much else.

5. You should take the commented code (called barley.c) and complete the program to calculate the total loss of barley. To do this use example code eleven to guide you.

6. In previous years the farmer has found that the aliens leave rings in his field and not circles. Write a second code that calculates the loss of barley for patterns of rings.

7. Because our farmer only wants to maintain a single codebase, combine the two programs that you have written above into a single code. Think about how the farmer might use this single code, also think about future events where the aliens might create both circles and rings.

8. Add a function to your code from 7 that calculates the area of the rectangular field. You should use the scanf() function to get the dimensions as input from the farmer.

9. Add a function to your code that will calculate the total loss of barley as a percentage of the barley that is harvested from the field.

10. Finally add a function that will calculate the monetary loss to the farmer.

Bonus questions

a. Create a Makefile for your codes
b. Upload your own codes to your git repo

Do not worry if you don't complete all of the above. The aim of this practical is to encourage you to write your own C code and become familiar with some of the common functions.