An Introduction to HPC and Scientific Computing

Wes Armour

Oxford e-Research Centre,
Department of Engineering Science

Aims and learning outcomes

The aims of this CWM are to introduce you to scientific computing and High Performance computing (HPC).

It’s more important that you pick up the basics of computing and programming during the week, because these are the building blocks for everything else.

This CWM isn’t designed to turn you into a world class HPC programmer, that that’s years.

This CWM is designed to give you the skills to continue to learn in this area and for you to have the ability to write your own computer codes and tackle basic problems.

Assessment for this course will focus on the final two practical sessions in the latter half of the week. The aim of the assessment is for you to demonstrate that you’ve picked up the basics from this course.

The assessment will be light because I’m keen for you to focus on the content rather than worrying about the assessment.

In all I hope you will find this a fun and interesting week long introduction to HPC and Scientific Computing!

Locations and Timetable

Locations

Lectures will be in LR6
Practical sessions will be in the Linux Lab

Timetable

09:30 - 10:30 Morning lecture
10:30 - 11:00 break
11:00 - 12:30 Morning practical

12:30 - 13:30 lunch

13:30 - 14:30 Afternoon lecture
14:30 - 15:00 break
15:00 - 16:30 Afternoon practical

Lectures will be delivered by Wes Armour, Ian Bush, Karel Adamek.

Practical's supervised by Wes Armour, Ian Bush, Karel Adamek, Ania Brown and Jan Novotny.

On-line feedback form: http://bit.ly/OXUNICWM please, please, please do complete ☺

Lectures

Monday - Here we have three lectures to begin with and finish with a practical session, this is because we'll need to introduce you to several different topics before you can complete a meaningful practical.

Morning lecture:	Introduction to computer architectures.
Morning lecture:	Introduction to the C programming language.
Afternoon lecture:	Introduction to Linux, compilers and build systems.

Tuesday

Morning lecture:	Using repositories and good coding practices.
Afternoon lecture:	A deeper dive into C programming.

Wednesday afternoon

Afternoon lecture:	How to multi-task on CPUs using OpenMP.

Thursday

Morning lecture:	An introduction to the CUDA programming language.
Afternoon lecture:	Scientific Computing using the CUDA programming language part one.

Friday

Morning lecture:	Scientific Computing using the CUDA programming language part two.
Afternoon lecture:	***Guest Lecture: Deep learning Demystified - Adam Grzywaczewski NVIDIA.***

Practical Sessions

Monday - Here we have one practical in the afternoon.

Afternoon Practical:	Linux, compiling C code and using Make.

Tuesday

Morning Practical:	Practical examples of using repositories for your projects.
Afternoon Practical:	Practical examples using the C programming language.

Wednesday Afternoon

Afternoon Practical:	Practical examples of using OpenMP on CPUs.

Thursday

Morning Practical:	Practical examples of the CUDA programming language.
Afternoon Practical:	Advanced examples of CUDA programming part one.

Friday

Morning Practical:	Advanced examples of CUDA programming part two.
Afternoon Practical:	Finishing up.

HLL, *.c, *.o and executables

We start with a hello.c code, this is a text file with our c code in it, its in human readable format.

We then compile this file (using gcc/icc) into an object file. Object files are in machine code that isn't (normally) executable.

We then use the compiler again to link these and produce an executable, for example a.out

You can change the name of a.out using the –o compiler flag.

The compiler normally performs both phases of compilation at the same time.

```
int square(int num) {

    return num * num;

}
```

*.c

```
push rbp #2.21
mov rbp, rsp #2.21
sub rsp, 16 #2.21
mov DWORD PTR [-16+rbp], edi #2.21
mov eax, DWORD PTR [-16+rbp] #3.18
imul eax, DWORD PTR [-16+rbp] #3.18
leave #3.18
ret #3.18
```

~= *.o

```
01110000011010001010100100100101001
00011111111111010101111001110011010
10101010011111111111010101001010101
10101010101010101001010101010100101
```

= ./a.out

Slurm

I tend to put all slurm commands in by slurm submission script.

The reason for this is that when I come to revisit the work at a later date, its easy to see what I've done and its easier to reproduce what I've done.

The websites on the right have lots of information about SBATCH commands.

https://slurm.schedmd.com/sbatch.html

http://www.arc.ox.ac.uk/content/slurm-job-scheduler

Using Linux

Linux provides many different ways to get the same task done because Linux has lots of small commands that can be put together.

The resources on the right can be quite helpful when working with linux.

Also google is your friend (in this instance).

http://www.arc.ox.ac.uk/content/introduction-linux

https://stackoverflow.com/

https://linuxconfig.org/bash-scripting-tutorial-for-beginners

https://stackoverflow.com/questions/5927369/recursively-look-for-files-with-a-specific-extension?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa