Tutorial on Unix/Linux

COMP2007

Geert De Maere
(Dan Marsden)
{Geert.DeMaere, Dan.Marsden}@Nottingham.ac.uk

University Of Nottingham
United Kingdom

2023

Goals for Today

- Overview to G52OSC labs
- Connecting to Linux School Servers
- Compiling and running your first program on School's servers

Plan

- 13 different tasks for the labs
  - Process creation & scheduling (1 – 4)
  - Threads and concurrency (5 – 7)
  - Shared memory, CPU affinity & process priority (8 – 10)
  - Files (11 - 13)
- Expectations:
  - 2 - 3 tasks session in the early labs
  - 1 - 2 tasks per session in the later labs

Goal

- Analyse Process scheduling in Unix/Linux
- Basic concepts for coursework
- Shared memory
- Core concepts of file systems

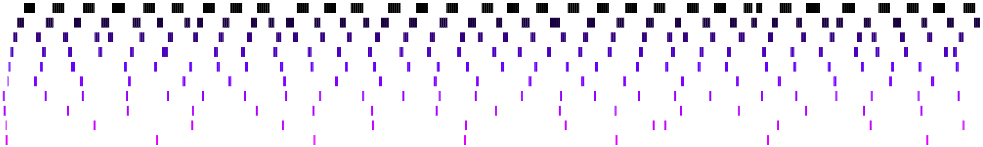


Figure: CPU Timings - Completely Fair Scheduler

Labs assessment

- Lab exercises are not assessed
- At least one (partial) question on the exam will be based on the labs
- Labs should help you better understand better the lectures

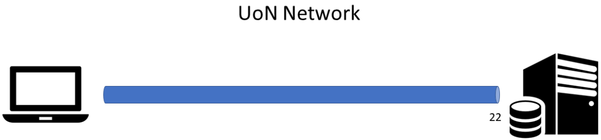
Servers at the school

- You will **always** have to compile and run your code on School's servers for both labs and coursework
- You **had** two options to connect to School's servers:
  - Physical servers (via SSH)
  - Virtualised environment (Nottingham Cloud)
- **Quiz**: Give one reason why you would use ssh over Nottingham Cloud, and one reason to use Nottingham cloud over ssh

Servers at the school

Connecting via SSH: From school PCs

- SSH is a protocol that allows us to remotely login to a machine (mostly Linux servers, and command-line login)
- For OSC, you should always use: *bann.cs.nott.ac.uk*



Connecting to School's servers

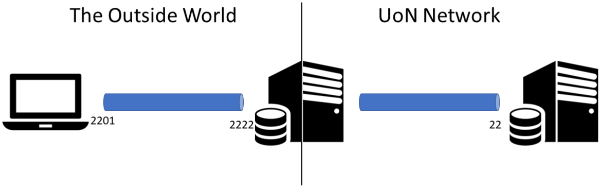
Connecting via SSH: From school PCs

- Use an ssh client (e.g. Putty is installed on CS computers)
- Connect to *bann.cs.nott.ac.uk* using your username (i.e. psXXX) and password.
- Your home directory is automatically mapped as H: on School's PCs.
- Write your programs using your favourite Windows editor (e.g. Notepad or Wordpad), and save it on your home directory (H:).

Servers at the school

Connecting via SSH: From your own laptop - Windows

- SSH is a protocol that allows us to remotely login to a machine (mostly Linux servers, and command-line login)
- For OSC, you should always use: *bann.cs.nott.ac.uk*



Connecting to School's servers

Connecting via SSH: From your own laptop - Windows

- Use an ssh client (e.g. Putty, WSL, Cygwin, CMD)
- Set up an ssh tunnel
- Connect to *bann.cs.nott.ac.uk* using your university user (i.e. psXXX) and password
- Writing Code:
  - Locally, copy remote
  - Remote using a Unix/Linux editor, e.g. vim, emacs, or nano
- Compile and run remote

Must-know Unix/Linux commands

```
pwd  - Print working directory
ls   - List files in current directory
cd <directory name> - move into a directory
mkdir <directory name> - Create a directory
exit - Leave SSH session.
mv <file name> <new file name/location>
- Move file/directories - Also change names of files.
cat <file name> - Display content of a file
chmod - Change permissions
man <Command name> - in-line manual.
```

Task: Create a new directory on your home folder for OSC Labs and make sure you are the only one with permissions to access it. Move into that directory and create a new source code file **hello.c**

Use Unix/Linux like a pro

- Use tab completion
- Use history of commands (up arrow)
- Use history for completion (page up)
- Use Linux text editors (e.g. nano, vim)
- More useful commands: tail, head, wc, grep
- Others: pipelines, redirections, scripting, ...

Compiling a C program

- Use ’gcc’ compiler
- Use -o <output filename> if you want to name your program. Otherwise, it will get ’a.out’ as name

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

- Add extra libraries if you need to at the end of the command:

```
$ gcc threadsExample.c -o threadsExample -pthread
```

Running a program

- Run your program, calling the executable file:

```
$ ./hello
```

- You can 'divert' the output of your program to a file using redirections:

```
$ ./hello > output.txt
```

- You can run your program multiple times (useful for the coursework):

```
$ for i in `seq 1 100`; do ./hello; done
```

- Check Moodle videos if you still have problems running your program

Good practice

Before you exit your session, Please ensure that you leave no processes running in the background on **bann.cs.nott.ac.uk**

- To see which processes you have running use "ps -ux"
- To kill all processes that you may have running in one go, please use *killall -u <user_name>*