1 Overview

This practical provides an overview of UNIX/Linux. It is intended as a "getting started" document for new users or for those who want to know "in a nutshell" what UNIX is all about from a practical user's perspective. It is also intended as the first presentation in a hands-on workshop that introduces programming on the ICHEC systems.

Once you have successfully logged in on the local system you can ssh to the VM cluster. ICHEC accounts have been set up for the purpose of this practical, sp## where ## is a number, which you have been assigned. For Linux and Mac users; from the command line you can log in as follows:

```
ssh username@sciprog.training.ichec.ie
```

Windows users will need to use MobaXterm to connect using the same host *sciprog.training.ichec.ie*, which can be downloaded from the MobaXterm website `https://mobaxterm.mobatek.net`.

Please change the password if you have not done so already when you log in first using *passwd* command. Each user's home folder is located in /home/.

For today's practical we will carry out all tasks in a folder called practical01. Type the commands below, if everything worked then you should see something like this $/home/sp1/practical01$

```
mkdir practical01    ## this is comment only; make directory
cd practical01       ## change directory
pwd                  ## show current directory
```

2 UNIX Shell

A shell is the interface between you and the UNIX system. The default shell is called "bash" (Bourne Again SHell). There are other shell environments, which one you use is just personal preference. A set of usable shells are listed in the file "/etc/shells". To view or list the contents of a file use *less e.g.*

```
less /etc/shells
```

Each shell can be customizable to suit your preferences. Each time a shell is started, system and user files are examined to setup the shell environment. For the bash shell the user file is ".bashrc" (in your home directory). To go to the home directory: *cd* $\sim$

If a file name starts with a dot (like ".bashrc"), these files are normally invisible. If you type *ls* in the home directory the file will not be present. The command *ls* lists the files stored within a directory.

3 UNIX Help pages

The UNIX help pages are a set of single pages per command or utility. Here a page refers to a file rather than a physical page. These pages are grouped into sections: section 1 contains the most relevant commands and section 3 contains help on C functions.

For instance *ls* lists the files within a directory. To view the UNIX help on *ls* type

```
man ls
```

You can view a command's man page if you know the command. Type 'q' to quit these man pages. However if you do not know exactly what you are looking for, you are stuck. The man pages can be searched using a keyword. To do this type

```
man -k <keyword>
man -k "list directory"
```

You can see that more than one command matches this keyword. The keyword is enclosed in double quotes because spaces are treated as delimiters. If using more than one word keywords they must be enclosed in double quotes.

Question

What argument to *ls* is required to list a file name starts with a dot (use *man*)? Try in your home directory. ls -a

4 Editing Files

In this section we will ensure everyone can create, save, open, edit, rename, and remove files. It will be left up to you to chose an editor. Emacs and vim are available. Basic Emacs and vi commands are available on page 7.

Please carry out the following steps:

1. Edit your .bashrc file. If not exist, create a new one.
   Bring up an editor window, using vi (*vi $HOME/.bashrc*), emacs (*emacs $HOME/.bashrc*). Unfortunately there is no X-windows server on the VM, so you cannot use multiple windows.
   Features include:
   (a) Either start a new file or open an existing one. You should be able to type in the edit window, delete and cut and paste with the mouse.
   (b) When saving the file it is important to give it the correct extension. If it is a C program then save with the extension ".c". If Fortran then save with extension ".f90".
   (c) Some editors detect what type of document you are working with and colour the text accordingly.
2. Add the following line:

   ```
   alias rm='rm -i'
   ```

3. Quit and save the .bashrc file.
4. Create the file xxx by typing 'touch xxx'. Verify it is there by typing 'ls'.
5. Delete the file using 'rm'.
6. Log out with 'exit' and back in again.

7. Do the same steps again create and delete xxx (Steps 4-6). Is there any difference?

   An interactive prompt would appear asking us to confirm if we really want to remove the file

5 Transferring Files

Before we move onto the next section we need to transfer files from BrightSpace to the VM. The first step is to copy any files from BrightSpace onto your own laptop. Then to copy them from the laptop to the VM, we need to have *scp*, which stands for secure copy. Again for UNIX and Mac users you should already have this. To copy *printing.c* to the VM use the command below. The copy is from the source to the destination, hopefully you can see that the file is being copied from your local machine to the VM.

```
scp printing.c sp1@sciprog.training.ichec.ie:~/.
```

For Windows users, MobaXterm has the facility for this. Now copy the same file *printing.c*.

6 Compilation

In this section you will work through the helloworld example, beginning with the very simplest compile and link. We have discussed multiple compilers but on the VM there are only the GNU compilers *gcc* and *gfortran*.

1. Both C and FORTRAN are compiled programming languages.
2. This means that the code is written in a human readable form and is converted into a machine readable form, by a compiler.
3. The machine readable form is saved to a different file which can then be executed. Running the program means to load the executable file into memory so that the CPU will execute the program commands.
4. After the program has finished the data stored in memory is not usually accessible. However information can be sent to the screen or to disk as the program is executing.
5. Below is a schematic of the compilation process. Code from different source files are transformed into object files. These object files along with external libraries are linked together to form an executable program.
6. External libraries contain sets of useful functions, usually written by third parties. The C maths library is an example.

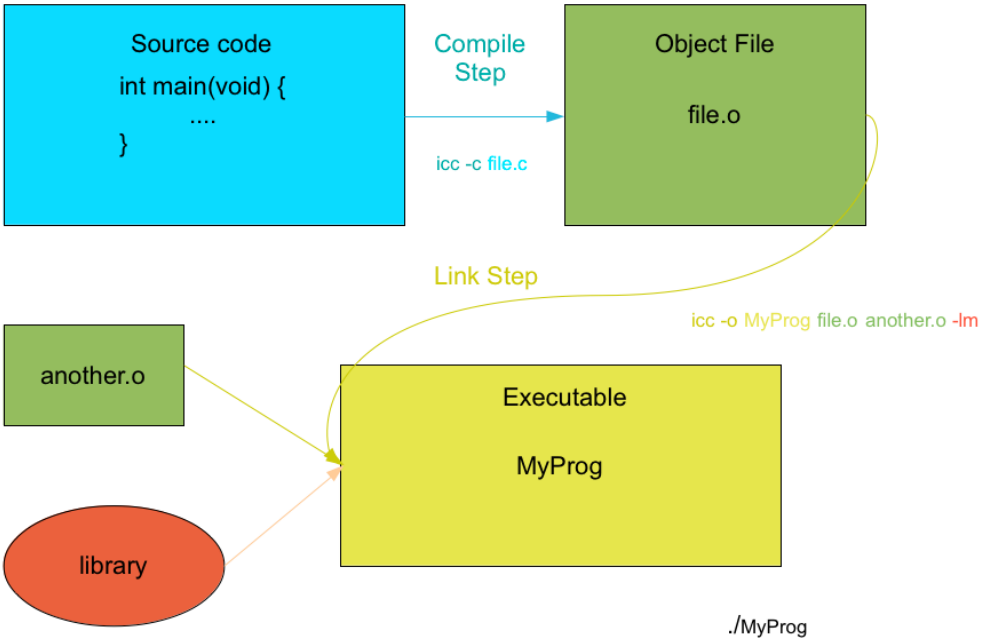

Figure 1. Schematic of the Compilation Process

6.1 Hello World

1. Create the file *helloworld.c* or *helloworld.f*90. You can use *emacs* or *vim* to edit the file, copy the lines of code below.

   The C program, helloworld.c:

```c
#include <stdio.h>
int main(void) {

    printf("Hello World\n");
    return(0);
}
```

   The FORTRAN program, helloworld.f:

```
!  THIS IS THE MAIN PROGRAM
   PROGRAM helloworld

   WRITE(*,*) 'Hello World'
   END
```

2. To compile the code use;

```
gcc -o helloworld helloworld.c

# or

gfortran -o helloworld helloworld.f90
```

3. To run the program type (the ./ tells the loader to look for helloworld in your current working directory; otherwise you may see "file not found" or a "Command not found" message)

```
./helloworld
```

7 Printing Syntax

We will cover how to print text to the screen in later lectures. However you can experience what you might expect to see and do.

7.1 Exercise

- Compile and run the printing code.
- Add the compile option $-save-temps$ and recompile.
- Do you see the additional files? .i file -> intermediate file, .s file -> assembler file, .o -> object file
- Compile and run the second program *scanning*.
- Notice that the program asks for you to input two positive integers.

- Again we will cover how to do this later but for many of the programs, for the course, there will be some user input.
- Try different numbers as well as reals numbers and characters and see what happens.

Quick Reference Guide

Getting Help

- **man** *program* (manual pages for *program*)
- **Google** *web searcher* (can find most information you need)

Unix Commands

- **ls** (list directory)
- **less** *file* (print a file to the screen)
- **cp** *file1 file2* (copy a file)
- **rm** *file* (delete a file)
- **mv** *file1 file2* (move or rename a file)
- **cd** *dirname* (change the current directory)
- **cat** (sends the file to standard output)
- **pwd** (print name of the current directory)
- **mkdir** *dirname* (create a directory)
- **rmdir** *dirname* (delete a directory)
- **exit** (quit the session)
- **passwd** (change password)
- **history** (list all commands given previously)
- **!stuff** (executes the last command that started with "stuff")
- **head** *file* (list ten first lines of the file)
- **tail -100** *file* (list the last hundred lines of the file)
- **tail -f** *file* (keeps listing the end of file. Handy for following an output file when lines are appended to it.)
- **grep** *stuff file* (print lines containing the word stuff from the file)
- **diff** *file1 file2* (lists file differences)
- **ls -la** > *file* (output of a command to a file)
- **ls -la** | **grep** "word" (chaining (piping) multiple commands)
- **tar cvf** t.tar t* (make a tar-file t.tar from all files whose names begin with t. You can also tar a directory.)
- **tar xvf** t.tar (extract all files from the tar-archive t.tar)
- **gzip** t.tar (compress file t.tar to save space)
- **gunzip** t.tar.gz (uncompress file t.tar.gz)

Computers

- **Sciprog:** *sciprog.training.ichec.ie*
  - Sciprog is a VM on Amazon.
  - It will only be accessible from the UCD eduroam network.
  - Feel free to use your own machine but of course you need to have a C or FORTRAN compiler
  - Bare in mind that the parallel programming course needs extra libraries which your laptop may not have, so getting used to UNIX maybe necessary.

File Transfer

- **scp** *computer1:file1 computer2:file2* (copy files from computer1 to computer2)
- An example of scp usage:
  scp *temp.txt username@sciprog.training.ichec.ie:.*
  (copies the file temp.txt (from current directory to sciprog) Because the directory in the target machine was not specified the file goes to the home directory of *username*.)
- Most ssh-clients also have a graphical file transfer program available. Ex: WinSCP in Windows.

Networking

- **ssh** *computer* (open a new secure session)
- In Windows you will need an ssh-program. Ex: MobaXterm
- In Linux use **ssh -X** *computer* or **ssh -Y** *computer* to enable X-connection
- In Windows to enable graphical X-windowing choose "Forward X11" from your ssh-program settings. You will also need a separate X-emulator program, e.g. XMing

Paging With less

- **less** file (print a file to the screen)
  - **[return]** (next line)
  - **[space]** (next screen)
  - **b** (previous screen)
  - **/stuff [enter]** looks for the next occurrence of "stuff"
  - **h** (list the commands of less)
  - **q** (quit the less program)

Unix Files Permissions

- **ls -l** (long list of files present. First column details the file permissions e.g. drwxr-xr- -)
  - Missing privileges are represented with a '-'.
  - **First** character: (file type: d = directory, - = file, l = link)
  - **2 - 4th** character: read/write/execute for **user**
  - **5 - 7th** character: read/write/execute for **group**
  - **8 - 10th** character: read/write/execute for **others**
- File permissions are altered by giving the chmod command and the appropriate octal code for each user type.
  Read = 4
  Write = 2
  Execute = 1
- **chmod 755** *file* (Full permission for the owner, read and execute access for the group and others.)
- **chmod +x** *file* (Make the file called filename executable to all users.)

Emacs Editor

- **emacs** *file* (start the emacs editor)
- **emacs -nw** *file* (emacs without X-windows)
- Notation **[Ctrl]-c** means: "hold down the Control key and press the c key"
- Moving: cursor keys and page up/down keys
- **[Ctrl]-x [Ctrl]-c** (quit and save)
- **[Ctrl]-x [Ctrl]-s** (save)
- **[Ctrl]-g** (interrupt an emacs command if you get stuck in the minibuffer)
- **[Ctrl]-h [Ctrl]-h** (Emacs help system)
- Other text editors are e.g. **nano**, and **vi**

VIM Editor

- **vi** *t.txt* (open file named t.txt)
- Two modes: insertion mode and command mode.
- **[ESC]** returns the editor to command mode
- **Inserting Text**
  - **i, I** (insert before cursor, before line)
  - **a, A** (append after cursor, after line)
  - **o, O** (open new line after, line before)
  - **r, R** (replace one char, many chars)
- **Deleting Text**
  - **x, X** (character to right, left)
  - **D** (to end of line)
  - **dd** (line)
- **Searching and Replacing**
  - **/w, ?w** (search forward/backward for w)
  - **/w/+n** (search forward for w and move down n lines)
  - **n, N** (repeat search (forward/backward))
  - **:s/old/new** (replace next occurrence of old with new)
  - **:s/old/new/g** (replace all occurrences on the line)
  - **:%s/old/new/g** (replace all occurrences in file)
  - **:%s/old/new/gc** (same as above, with confirmation)
- **Undoing**
  - **U** (undo changes on current line)
  - **u** (undo last command)
- **Quitting**
  - **:x or :wq** (exit, saving changes)
  - **:q** (quit (unless changes))
  - **:q!** (quit (force, even if unsaved))

System Status

- **ps** (process status)
- **top** (continuous process status)
- **uptime** (show the load of the computer)
- **who** (list logged-in users)
- **whoami** (display current user)
- **date** (print data & time)
- **finger** *user* (gives information about user)
- **df -kh** (disk status in human readable units)
- **du -kh** (disk space used by a directory)
- **qsub**, **qstat**, **qdel** (submit, get status of, and cancel batch jobs in torque)
- **lfs quota -u your_username /ichec/home** *project* (List your user quota)

Finding files and text within files

- **find** */ -name fname* (Starting with the root directory, look for the file called fname)
- **find** */ -name "*fname*"* (Starting with the root directory, look for the file containing the string fname)
- **grep** *textstringtofind /dir* (Starting with the directory called dir , look for and list all files containing textstringtofind)

Program Development

- Compilers on ICHEC machines (Fortran, C): GNU, Intel
- An example of compiling a program with gcc:
  cc -o prog -fast prog.c
  Run the program: ./prog

How to contact ICHEC

- WWW homepage: `www.ichec.ie`
- Staff: `www.ichec.ie/staff`
- Helpdesk: `https://www.ichec.ie/academic/national-hpc/user-support`
- Dublin Office Address:

  ICHEC

  7th Floor Tower Building

  Trinity Technology & Enterprise Campus

  Grand Canal Quay

  Dublin 2, Ireland

  T: +353 1 5241608 F: +353 1 7645845
- Galway Office Address:

  ICHEC

  IT302, IT Building

  NUI Galway

  Galway City, Ireland

  T: +353 91 495305 F: +353 91 495573