**Workshop Bash at Utrecht University (UU) – Notes**

HPC (high performance computer) is the supercomputer (but any machine in the cloud would do too). There are 20 in UU, but one can access also Gemini, or nationwide Snellius and SURF Research cloud.

Choosing which computer to use depends on requirements (CPU, RAM …), softwares, security or support level. Faculty IT supports, but for us we might have to rely on SURF.

For Primers: git and bash are skills to work on supercomputers.

Basics for supercomputers follows on Wednesday.

Coffee brake at 11:00, Lunch at 12:30, last session is 13:15-14:30.

Jelle is trainer with Dawa and Shiva. Lukas is a scientific programmer / developer.

**Introduction to linux command line**

A shell is where you type commands to do things. You type them in a prompt. A shell is not always necessary, as there are portals like Jupyter or RStudio that allows you to click your way around, without knowing the syntax.

However most supercomputers are linux based and can only be operated using a shell. But working with GIT will make it easier. Bash is the default shell on linux systems. Powershell or Zshell are also often used.

**Exercise:**

Bilologist has 1520 samples, measuring relative abundance of 300 proteins, she needs to runs 1520 files through an imaginary program called goosthas. We explore how to automate with bash the 1520 analysis that the biologist must perform. In order to do so we:

* Navigate files/directories
* Create files/directories
* Check length of files
* Chain of command
* Retrieve files
* Iterate files

Open bash terminal: our computer has a file system of folders (directories) and files. We learn commands related to this first such as **pwd** (present working directory). This is also explained in the textbook Module2 (<https://utrechtuniversity.github.io/workshop-introduction-to-bash/modules/module2.html>) on the website of UU, which also links to all other 6 modules that compose today’s material. Once you digit this command a **/** (slash) appears that is called the root directory. Another command is **ls** (list) that delivers the content of a directory. This also allows options such as **ls -F**, which allows highlighting file types, for instance executables … but to see all options you can digit **ls --help**. Notice that the **man ls** (manual) command does not work in windows.

**clear** removes all the text generated so far in the log

if you type **ls Documents** you see what is in the Documents folder (provided you have such folder in your directory).

To get into a folder you use the **cd** command (such as **cd Documents**). And to go back to your home directory use **cd ~,** while going back to previous directories you type **cd ..** while going back two directories at once is done using **cd../..** One can also get into a subdirectory at once typing for instance **cd Documents/’My Music’**

So far we have seen the prompt **$** the command such as **ls**, options such as **-F** and the argument that comes after the **/** such as a directory name. Remember to always leave a blank space in between.

Second part is about moving an copying files and directories. Here an example of the code we worked with today: A screen shot of a computer program

AI-generated content may be incorrect.

Nano is an editor that you can use. To exit is you go to **ctrl x**. With **control o** you save the file. When you get stuck use **ctrl c**

A computer screen shot of a program

AI-generated content may be incorrect.

**mv** is move … so **mv draft.txt thesis/** basically moves the txt file to directory thesis.

**cp** copies … so **cp draft.txt draft2txt** creates a copy if the txt file

for directory copying you must add the **-r** option (in order to copy all subfolders**).** For instance: **$ cp thesis/ newtehsis/ -r**

Below we show some examples of copying commands. We also show how to delete them and how to use **\*** and **?** to do this more quickly. A screen shot of a computer program

AI-generated content may be incorrect.

See below that the **mv** (move) command with directories both moves directories to other destinations but can be also used to rename them. See below:A screenshot of a computer program

AI-generated content may be incorrect.

**Additional commands**

We now discuss some commands that can be used to explore the files in our directories and their content.

**wc** = word count: A computer screen shot of a program

AI-generated content may be incorrect.

Sorting file in the display (not in the file) is possible and you can store that output too: A screenshot of a computer screen

AI-generated content may be incorrect.

**-n** is the number of lines that you want to display. You can also combine these commands: A screen shot of a computer

AI-generated content may be incorrect.

Adding more commands is also possible: A screen shot of a computer screen

AI-generated content may be incorrect.

The difference between the **>** and **|** is that the first creates a new file, while **|** leaves the output only temporarely in the shell.

The name of **wc**, **sort** and **head** is FILTERS. This is a specific class of commands.

These can be useful for file inspection of missing info: A screenshot of a computer screen

AI-generated content may be incorrect.

The Z files have death info, how many are there? Only TwoA screen shot of a computer

AI-generated content may be incorrect.

**After lunch**

We now explore more capabilities of the terminal and use these to explore the difference between sequentailly running a program or in parallel.

We can make loops: A black screen with yellow text

AI-generated content may be incorrect.

You can also create a code that does this A computer screen shot of text

AI-generated content may be incorrect.

The **&** after the bash command allows running processes in parallelA screen shot of a computer

AI-generated content may be incorrect.

Notice however that it is your responsibility to know if a code needs running in parallel or sequentially.

**Now we do to the supercomputer**

Instructions for this part can be found at:

<https://utrechtuniversity.github.io/workshop-introduction-to-bash/modules/module6.html>

Log in into super computerA screen shot of a computer

AI-generated content may be incorrect.

This is what you see: 

We first download data from the internet and put it on the supercimputer

Check that you have a ssh key to avoid entering your password all the time: A screenshot of a computer screen

AI-generated content may be incorrect.

Now you have the downloaded zip file without needing a passwrod: A screenshot of a computer screen

AI-generated content may be incorrect.

**Day 2: GIT**

11:00 Break of 15 minutes

12:30 lunch

**Version Control**

Keep track of different versions of the same thing. We use GitHub, but gitlab and codeberg would work the same.

There are some practical files under <https://github.com/dianaiusan/recipe-book>

This is a repository. It looks nice and describes your package in a clear way. You see the main and the number of commits (versions) . When you one those you will see in green what was added and in red what was deleted.

Anything you put on github is public, so do not put any data.

A fork is a copy of the repository that becomes yours. When you click on fork and accept the copy, you create <https://github.com/mmo230/recipe-book>

If you have own code and don’t fork you need to go to either a R or Python template and **use own repository**. The R-template can also be used for Stata. If you use C++ you better use the template of python.