High Performance
Computing &
Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

LUXEMBOURG
LET'S MAKE IT HAPPEN

HPC School - Beginner

S1-2 - Don't fear the command line

UNIVERSITÉ DU
LUXEMBOURG

Before we start

Objectives:

- get acquainted to the linux command line interface (CLI)
- be able to manipulate the file system
- be able to decrypt complicated commands

Prerequisite: you should be able to connect to the HPC cluster

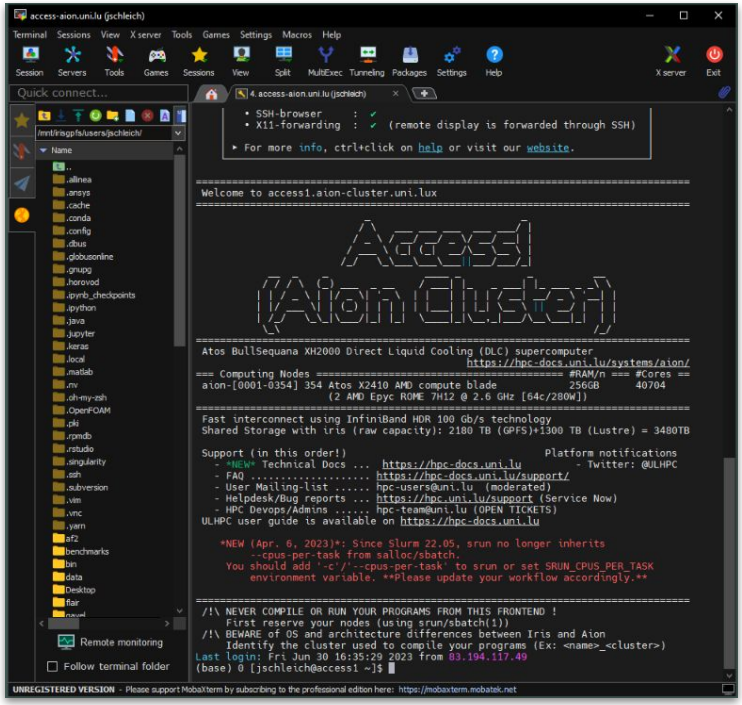
A little bit of history

- Computers before the early 80s'
  - room sized, expensive
  - as powerful as a modern scientific calculator
  - multi user
  - central computer / terminal model
- Philosophy
  - use resources as efficiently as possible
  - textual human/machine interface (shell)
  - small specialized programs

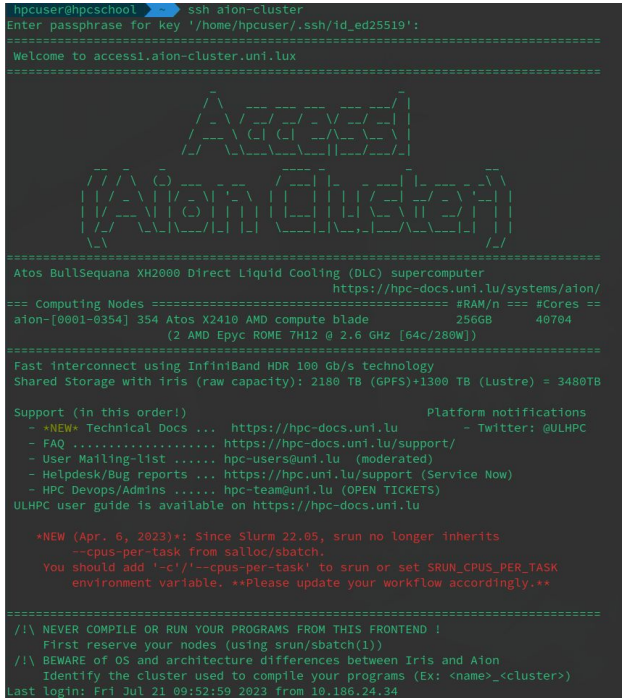
Similarities with the modern HPC and Cloud ethos

Connect to the HPC cluster

Mobaxterm



CLI



The command prompt

The bash shell

```
0 [aglad@access1 ~]$
```

- previous command exit status
- username
- hostname
- current directory
- command prompt

```
0 [aglad@aion-0235 git](871665 1N/T/1CN)$
```

- (job id node count/task count/task per node)

The linux file system

- Tree structure starting at **/**
  - No drives as in windows (c:, ...)
- Strong conventions
  - System directories
    - **/etc** - configuration files
    - **/bin** - built-in programs
    - **/usr** - 'user system resource'
      other installed programs, libraries, ...
    - **/home** - users personal directories
  - Program installation follows conventions
    - difficult to know what is installed
    - the system able to provide **completion**
- Files and directories
  - **~** - shorthand for **/home/users/<user_name>**
  - **.** - current directory
  - **..** - parent directory
  - **.filename** - hidden file/directory

```
total 512
drwx------ 18 hpcuser hpcuser   4096 Sep  7 15:25 .
drwxr-xr-x  5 root    root      4096 Jul 21 09:41 ..
-rw-------  1 hpcuser hpcuser     59 Jul 21 19:26 .bash_history
-rw-r--r--  1 hpcuser hpcuser     21 Jan  9  2022 .bash_logout
-rw-r--r--  1 hpcuser hpcuser     57 Jan  9  2022 .bash_profile
-rw-r--r--  1 hpcuser hpcuser   3824 Jan 14  2022 .bashrc
drwxr-xr-x 13 hpcuser hpcuser   4096 Aug  7 09:55 .cache
drwxr-xr-x 14 hpcuser hpcuser   4096 Aug  7 09:54 .config
-rw-r--r--  1 hpcuser hpcuser      0 Jul 21 11:12 data.dat
drwxr-xr-x  2 hpcuser hpcuser   4096 Aug  7 09:51 Desktop
-rw-r--r--  1 hpcuser hpcuser   4855 Oct 30  2017 .dir_colors
drwxr-xr-x  2 hpcuser hpcuser   4096 Aug  7 09:51 Documents
drwxr-xr-x  2 hpcuser hpcuser   4096 Aug  7 09:51 Downloads
```

- Access rights
  - user - group
  - **d[rwx][rwx][rwx]**
    - directory [user][group][other]
    - read - write - execute/navigate

Anatomy of a command

program [flags]... [arguments]...

- flags = options that change the behavior of the program
  - not sensitive to order
  - long flags - more readable, annoying to type
    - start with -- e.g. `--name`
    - can have parameters e.g. `--format=long --ignore foo`
  - short flags - condensed, difficult to parse
    - start with - e.g. `-l -a`
    - can be combined - e.g. `-la`. The order of the tags is not relevant `-la = -al`
    - can also have parameters - e.g. `-I foo`
- arguments = parameters of the program
  - positional
  - number of arguments depends on the program
- example - all these commands are equivalent
  - `$ls -la`
  - `$ls -a -l`
  - `$ls --format=long -a`

A little bit of help

- Manual page ⟵ *more comprehensive*
  - `man <command>`
    - e.g. `$man ls`
- Help flag ⟵ *when a man page does not exist*
  - `<command> --help`
    - e.g. `$ls --help`

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

 Manual page ls(1) line 1 (press h for help or q to quit)
```

Now, you do it!

- What does the **`echo`** command do?
- What does the **`which`** command do?

Lëtz build ourselves a little playground

```
$ cd

$ git clone https://gitlab.uni.lu/hlst/hpc-school-for-beginners.git

$ ls -R ~/hpc-school-for-beginners

$ cd hpc-school-for-beginners/CLI
```

Navigating through the file system

Tab autocompletes paths and your commands

- **pwd** - print working directory
  - show the full path of the current directory
  - useful to know where you are
- **ls** - list
  - list the files and directories in the current directory
  - add a path in argument to show the content of another directory
  - **-a** flag shows hidden files and directories
  - **-l** formats the output and shows access right
- **cd** - change directory
  - **cd** with no argument returns you to your home dir
  - **cd /<path>** - go the the indicated absolute path
    - e.g. **$ cd /home/users/hpcuser/foo/**
  - **cd ./<path>** or **cd <path>** - go to the relative path
    - e.g. **$ cd ./foo** then **$ cd nestedFoo**
  - **cd ..** - go to the parent directory
    - e.g. from ~/foo/nestedFoo - **& cd ../../dir**

```
[hpcuser@hpcschool git]$ pwd
/home/hpcuser/git
```

```
[hpcuser@hpcschool CLI]$ ls -la
total 24
drwxr-xr-x 6 root root 4096 Sep  8 14:27 .
drwxr-xr-x 4 root root 4096 Sep  8 14:27 ..
drwxr-xr-x 2 root root 4096 Sep  8 14:27 docs
```

Now, you do it!

- go to your home directory
- from there, go to the tutorial directory
  - **hpc-school-for-beginners/CLI**
- go back a level then to the docs directory

Navigating through the file system

- **pwd** - print working directory
  - show the full path of the current directory
  - useful to know where you are
- **ls** - list
  - list the files and directories in the current directory
  - add a path in argument to show the content of another directory
  - **-a** flag shows hidden files and directories
  - **-l** formats the output and shows access right
- **cd** - change directory
  - **cd** with no argument returns you to your home dir
  - **cd /<path>** - go the the indicated absolute path
    - e.g. **$ cd /home/users/hpcuser/foo/**
  - **cd ./<path>** or **cd <path>** - go to the relative path
    - e.g. **$ cd ./foo** then **$ cd nestedFoo**
  - **cd ..** - go to the parent directory
    - e.g. from ~/foo/nestedFoo - **& cd ../../dir**

```
[hpcuser@hpcschool git]$ pwd
/home/hpcuser/git
```

Tab autocompletes paths and your commands

```
[hpcuser@hpcschool CLI]$ ls -la
total 24
drwxr-xr-x 6 root root 4096 Sep  8 14:27 .
drwxr-xr-x 4 root root 4096 Sep  8 14:27 ..
drwxr-xr-x 2 root root 4096 Sep  8 14:27 docs
```

```
[hpcuser@hpcschool docs]$ cd
[hpcuser@hpcschool ~]$ pwd
/home/hpcuser
[hpcuser@hpcschool ~]$ cd hpc-school-for-beginners/
[hpcuser@hpcschool hpc-school-for-beginners]$ cd CLI/playground
[hpcuser@hpcschool playground]$ pwd
/home/hpcuser/hpc-school-for-beginners/CLI/playground
[hpcuser@hpcschool playground]$ cd ../docs
[hpcuser@hpcschool docs]$ pwd
/home/hpcuser/hpc-school-for-beginners/CLI/docs
```

Executing programs and scripts

- Built-in and installed software
  - Built-in commands come with the shell
  - Installed software
    - via package manager or install scripts
    - copies files according to conventions
  - The OS is aware of their existence/location
    - just type their name (e.g. `ls`, `cd`)
    - the tab key proposes completion (e.g. `l + tab` -> all executables starting with 'l')
- Scripts and non installed software
  - Launcher scripts (.sh), precompiled software from archives
  - The OS is usually **not** aware of their existence
    - must be executable
    - called using its *path/name* (either absolute (starts with `/`) or relative (starts with `./`)

Now, you do it!

- Go to `~/hpc-school-for-beginners/CLI`
- Run the backup script located in `playground/scripts/backup.sh` with the file `playground/files/important.txt` as an argument
- Check the content of the `playground/backup` directory

Manipulating files (1/2)

- `mkdir` - make directory
  - create a directory at the designated path
    - `$ mkdir test`
    - `$ mkdir i_dont_exist/test` -> The command fails because the i_dont_exist directory does not exist.
    - `$ mkdir -p i_dont_exist/test` -> Recursively creates the directories if they do not exist.

- `cp` - copy
  - copy a file - `cp <source> <destination>`
    - `$cp ./dir/file.txt file(copy).txt`
    - `$ls` -> you should see test.txt
  - copy a directory - use the `-r` flag
    - `$cp -r ./dir ./foo/nestedFoo`
    - `$cp -r ./dir ./foobar`
      The command fails. The foobar directory doesn't exist.
  - copy files using a pattern
    - `$cp  dir/* foo`
      copies all files in dir to foo
    - `$cp  dir/*.txt foo`
      copies all files ending in .txt in dir to foo

Now, you do it!

- Create a 'manual backup' directory in the CLI directory
- Make a backup of `~/hpc-school-for-beginner/CLI/playground/temp/experiment.out` in the 'manual backup' directory

Manipulating files (2/2)

- `rm` - remove file
  - There is no bin. Deleted files **cannot** be recovered.
  - delete a file
    - `$ rm ~/hpcschool/foo/data.dat`
  - delete a directory
    - `$ rm -r ~/hpcschool/foo/nestedFoo`
  - force deletion -
    - `$ rm -f -r ~/hpcschool/foo`
- `mv` - move file
  - move a file/directory (cut and paste)
    - `$ mv test.txt foo/test.txt`
  - rename a file/directory = moving a file to the same directory
    - `$ mv foo/test.txt foo/temp.txt`
  - move and rename a file
    - `$ mv foo/temp.txt ./test.txt`

up and down arrows navigate through recent commands

Now, you do it!

- Delete the `~/hpc-school-for-beginner/CLI/playground/temp` directory and its content
- Move the backup of experiment.out to the `hpc-school-for-beginners/CLI/playground/files/experiment/data` directory and rename it `test001.com`

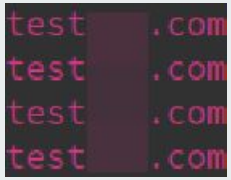
Reading and writing files (1/2)

ctrl-r allows to search your command history

- cat - concatenates files and write to the standard output
  - `$ cat <filename>`
  - e.g. `$ cat dir/data.csv`
- less - reading longer files
  - `$ less <filename>`
  - e.g. `$ less dir/data.csv`
  - scroll with arrows/page up/page down
  - quit with q
- tail - show the last lines of a text file
  - `$ tail <filename>`
  - `$ tail -n 25 <filename>` - specify the number of displayed lines
  - `$ tail -f <filename>` - follow new lines
- chaining commands - the | operator
  - allow to pass the output of a command to the next one
  - `$ ls -la /usr/bin | less`
- grep - filtering utility
  - `$ ls /usr/bin | grep update`
  - `$ grep pattern filename`

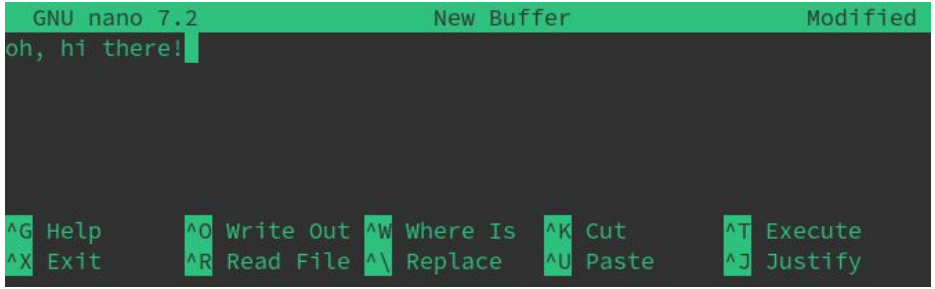
Now, you do it!

- Launch `playground/scripts/tailMe.sh&` and follow the output file (tailMe.out)
- Find all files that contain data about Methylene (with a capitalized M)
- Find all files that contain multi part jobs (like test101.com. You should find 4 files. Bonus points if you manage to get the following output



Reading and writing files (2/2)

- > and >> - write standard output to a file
  - `& ls /usr/bin > all_bins.txt`
    - creates or opens in **overwrite** mode the all_bins.txt file and write the output of ls /usr/bin in it
  - `& ls -la /usr/bin >> all_bins.txt`
    - creates or opens in append mode the all_bins.txt file and write the output of ls -la /usr/bin in it
- the nano text editor
  - `$ nano [filename]`
  - basic usage
    - write with your keyboard
    - move the cursor with the arrow keys
  - shortcuts
    - at the bottom of the screen
    - accessible via ctrl + <key>
    - e.g. save: ctrl + o; quit: ctrl + x



Changing permissions

chmod - **ch**ange file **mod**e bits
allows you to change access rights on your files and directories

```
drwxr-xr-x  4 root    root    4096 Sep  8 14:27 .
drwx------ 19 hpcuser hpcuser 4096 Sep  8 14:30 ..
drwxr-xr-x  6 root    root    4096 Sep  8 14:27 CLI
drwxr-xr-x  8 root    root    4096 Sep  8 14:27 .git
-rw-r--r--  1 root    root    6213 Sep  8 14:27 README.md
```

Why?

- give/restrict access to other people
- make some files executable

$chmod 744 <filename>

- first number for yourself, second for your group, third for everyone else
- octal mode
  - 1 = execute
  - 2 = write
  - 4 = read
- for directories
  - execute allows to cd into
  - write allows to create/delete files
  - read allows to see the content of the directory

sum the numbers to adjust the rights

- 7 = 1+2+4 = x + w + r
- 4 = read only

Now, you do it!

- **make CLI/playground/files/surprise.txt readable only to you**
- **make CLI/scripts/helloWorld.sh executable and run it**

Moving data

- Rsync is a utility that allows to synchronize data between machines
  - upload/download files
  - synchronize files between servers
  - resume interrupted transfers

- Push data
  - `&rsync -azvu <source> [user@]<host>:<destination>`
  - `&rsync -azvu data_directory aion-cluster:my_data`
- Pull data
  - `&rsync -azvu [user@]<host>:<source> <destination>`
  - `&rsync -azvu aion-cluster:my_data data_directory`
- Flags
  - a - archive mode (recursive, copies files, rights, links, ...)
  - z - compress data during transfer (speeds up transmission)
  - v - verbose (display what is going on)
  - u - update (skip files that are newer on the receiver)
  - P - progress bar (monitor big transfers)

Now, you do it!

- copy the content of the CLI/docs directory to your machine to get this presentation and a command line interface cheat sheet pdf

run rsync from your laptop, not from the HPC

Final Boss

```
$ man bash > tmp.dat
$ cat tmp.dat | grep -i bash | wc -l
```

Find out what these commands are doing. Don’t run them yet!

Final Boss

```
$ man bash > tmp.dat
$ cat tmp.dat | grep -i bash | wc -l
```

Find out what these commands are doing. Don't run them yet!

- first command
  - redirect the content of the man page command for the bash program to the tmp.dat file
- second command
  - display the content of the tmp.dat file (cat) and
  - pipe the result to grep. Only keep the lines that contain 'bash' while ignoring the case and
  - pipe the output to wc which will count the number of lines (-l flag)
- summary: count the number of lines containing 'bash' (case insensitive) in the man page of the bash program

Final Boss - Phase 2

```
$ export HPL_VERSION=2.3
$ wget --continue http://www.netlib.org/benchmark/hpl/hpl-${HPL_VERSION}.tar.gz
$ tar xvzf hpl-${HPL_VERSION}.tar.gz
```

Find out what these commands are doing? Don't run them yet!
You might need the help of google on this one!

Final Boss - Phase 2

```
$ export HPL_VERSION=2.3
$ wget --continue http://www.netlib.org/benchmark/hpl/hpl-${HPL_VERSION}.tar.gz
$ tar xvzf hpl-${HPL_VERSION}.tar.gz
```

Find out what these commands are doing? Don't run them yet!
You might need the help of google on this one!

- set an environment variable
- download a file from netlib.org. The name of the file depends on the value of the environment variable that has been set previously. If the file was partially downloaded, continue instead of redownloading everything.
- extract the files (x) from a gzip archive (z) in the hpl-2.3.tar.gz file (f) and show the logs (v)

Final Boss - Final form

Make `~/hpcschool/data/runme.sh` executable and run it.

What did it do? How can you get rid of it? The script might contain clues and you have all the keys…

Final Boss - Final form

Make `~/hpcschool/data/runme.sh` executable and run it.

What did it do? How can you get rid of it? The script might contain clues and you have all the keys…

- Too many files to be deleted one by one. Maybe they all have a pattern in common?
- It seems that you don't have the rights to remove files in this directory

Final Boss - Final form

Make `~/hpcschool/data/runme.sh` executable and run it.

What did it do? How can you get rid of it? The script might contain clues and you have all the keys…

- Too many files to be deleted one by one. Maybe they all have a pattern in common?
- It seems that you don't have the rights to remove files in this directory

- Regain write rights on the directory `$chmod 700 ~/hpcschool`
- All files finish with a 1. Delete them using a pattern. `$rm -f ~/hpcschool/*1`

Never trust random scripts and commands found on the internet. Try to understand them first!
Your rights are limited and you cannot really hurt the HPC.
You could easily lose you data however.