McGill initiative in Computational Medicine

Intro to Unix

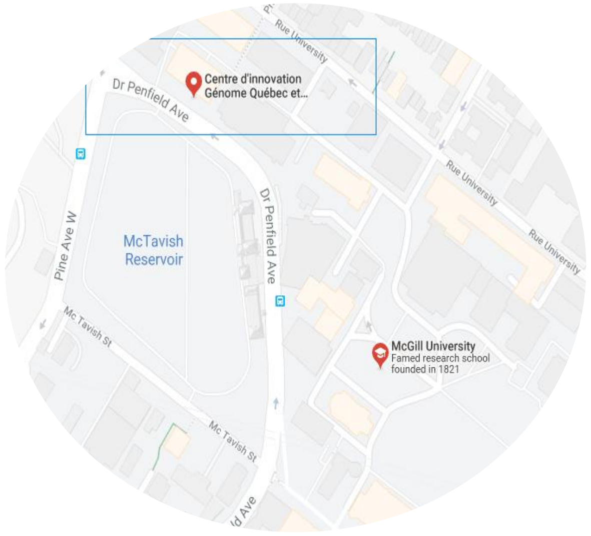
Workshop Lead: Georgi Merhi
Workshop Facilitator:
Slides source: Larisa Morales Soto - Edited by: Georgi Merhi

Date: February 6th, 2025

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

McGill initiative in Computational Medicine

Mission statement: deliver quality workshops designed to help biomedical researchers develop the skills they need to succeed.



Location: 740 Dr. Penfield Avenue, Montreal, Quebec

Scan the QR code to sign up for our **mailing list**

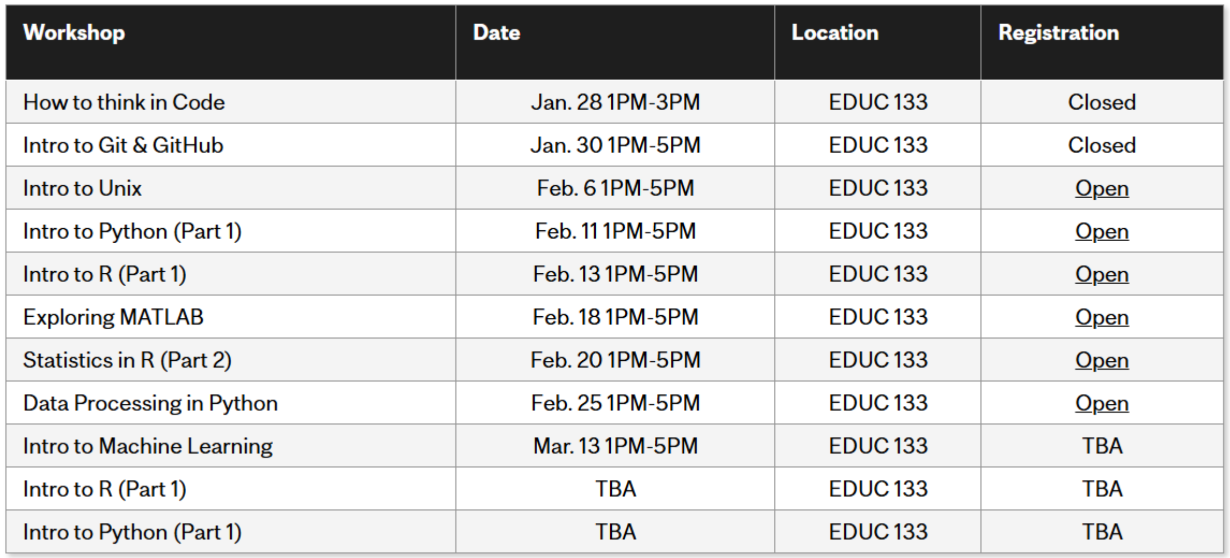
Contact: workshop-micm@mcgill.ca

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

McGill initiative in Computational Medicine

Winter 2025 Workshop Series

Winter Series



Workshop	Date	Location	Registration
How to think in Code	Jan. 28 1PM-3PM	EDUC 133	Closed
Intro to Git & GitHub	Jan. 30 1PM-5PM	EDUC 133	Closed
Intro to Unix	Feb. 6 1PM-5PM	EDUC 133	Open
Intro to Python (Part 1)	Feb. 11 1PM-5PM	EDUC 133	Open
Intro to R (Part 1)	Feb. 13 1PM-5PM	EDUC 133	Open
Exploring MATLAB	Feb. 18 1PM-5PM	EDUC 133	Open
Statistics in R (Part 2)	Feb. 20 1PM-5PM	EDUC 133	Open
Data Processing in Python	Feb. 25 1PM-5PM	EDUC 133	Open
Intro to Machine Learning	Mar. 13 1PM-5PM	EDUC 133	TBA
Intro to R (Part 1)	TBA	EDUC 133	TBA
Intro to Python (Part 1)	TBA	EDUC 133	TBA

https://www.mcgill.ca/micm/training/workshops-series

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

Outline

- Module 1 0
  - UNIX operating system
  - Filesystem structure
  - File management operations
  - Basic commands
- Module 2
  - Pipes and redirects
  - Variables
  - Pattern matching
  - Text processing
- Module 3
  - Shell scripting
  - Control structures
  - High performance computing (HPC)

Materials: https://github.com/McGill-MiCM/MiCM_Winter2024

Module 1

- UNIX operating system
- File system structure
- File management
- Basic commands

Dennis Ritchie (standing) and Ken Thompson (seating) at Bell Labs, 1960's

UNIX Operating system

- General purpose and interactive
- Multiuser
- Resource-efficient
- Hierarchical file system
- Compatibility (files, devices, processes)
- Time sharing

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

Parts

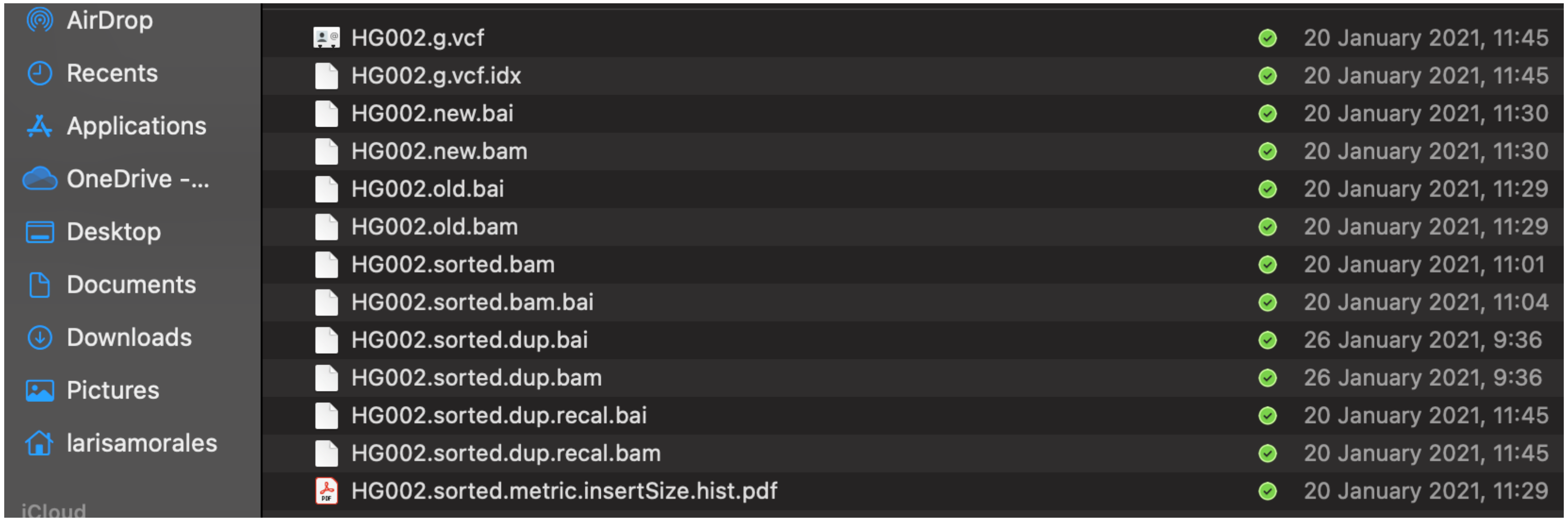
The kernel

- The core of the operating system
- Manages processes' time, memory and resources
- Handles file storage
- Responds to system calls

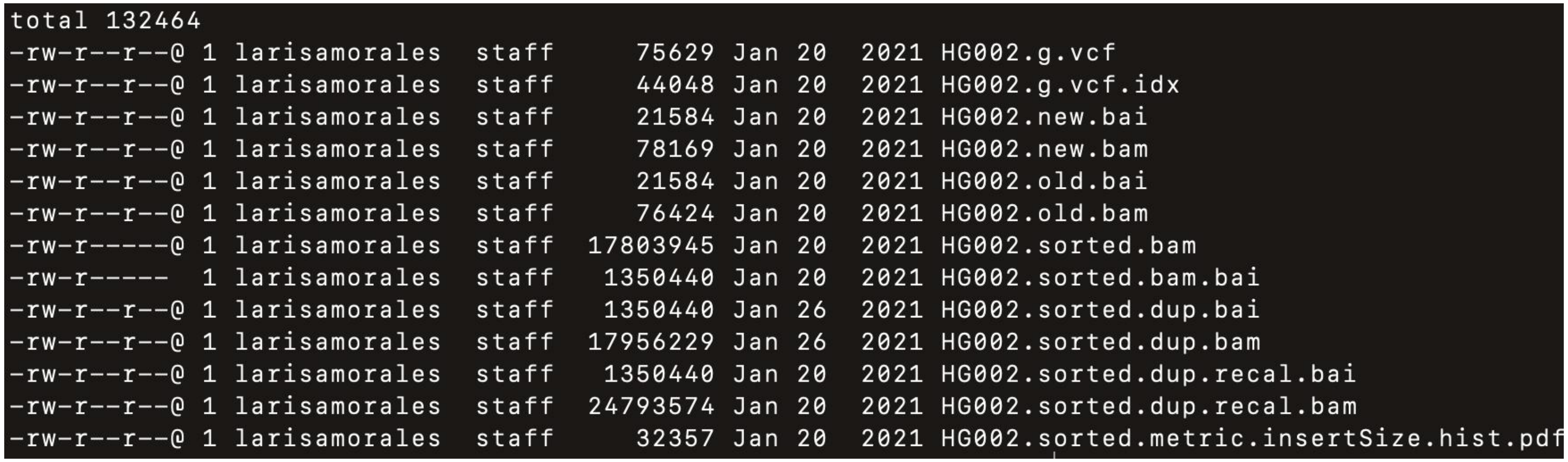
The shell

- Let's the user communicate with the kernel
- Command line interpreter (CLI)
- Executes the instructions requested by the user (commands)

Files

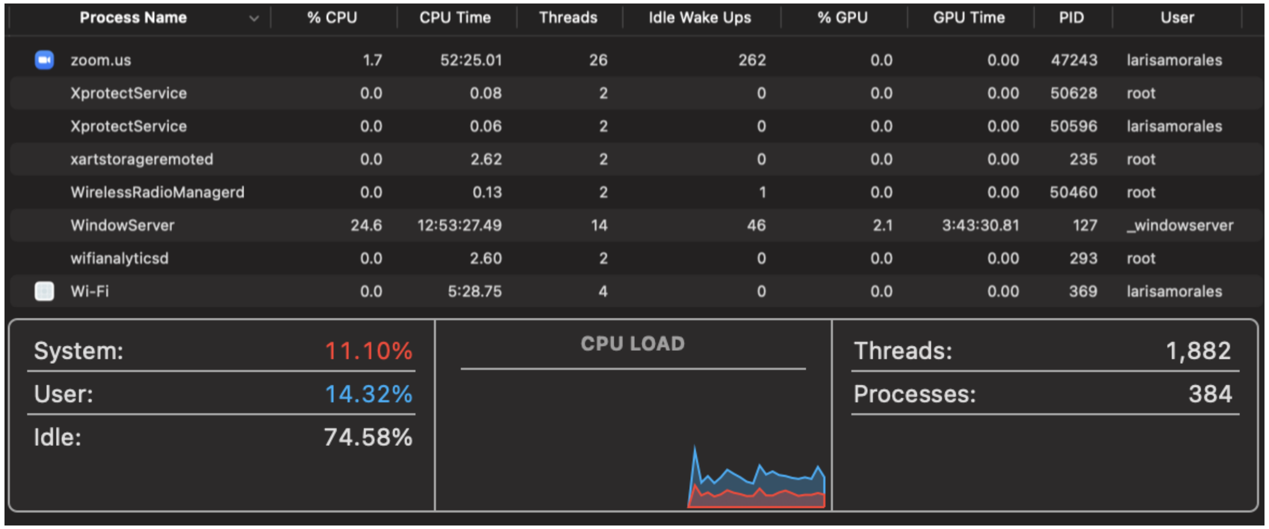


HG002.g.vcf	20 January 2021, 11:45
HG002.g.vcf.idx	20 January 2021, 11:45
HG002.new.bai	20 January 2021, 11:30
HG002.new.bam	20 January 2021, 11:30
HG002.old.bai	20 January 2021, 11:29
HG002.old.bam	20 January 2021, 11:29
HG002.sorted.bam	20 January 2021, 11:01
HG002.sorted.bam.bai	20 January 2021, 11:04
HG002.sorted.dup.bai	26 January 2021, 9:36
HG002.sorted.dup.bam	26 January 2021, 9:36
HG002.sorted.dup.recal.bai	20 January 2021, 11:45
HG002.sorted.dup.recal.bam	20 January 2021, 11:45
HG002.sorted.metric.insertSize.hist.pdf	20 January 2021, 11:29



```
total 132464
-rw-r--r--@ 1 larisamorales  staff     75629 Jan 20  2021 HG002.g.vcf
-rw-r--r--@ 1 larisamorales  staff     44048 Jan 20  2021 HG002.g.vcf.idx
-rw-r--r--@ 1 larisamorales  staff     21584 Jan 20  2021 HG002.new.bai
-rw-r--r--@ 1 larisamorales  staff     78169 Jan 20  2021 HG002.new.bam
-rw-r--r--@ 1 larisamorales  staff     21584 Jan 20  2021 HG002.old.bai
-rw-r--r--@ 1 larisamorales  staff     76424 Jan 20  2021 HG002.old.bam
-rw-r-----@ 1 larisamorales  staff  17803945 Jan 20  2021 HG002.sorted.bam
-rw-r-----  1 larisamorales  staff   1350440 Jan 20  2021 HG002.sorted.bam.bai
-rw-r--r--@ 1 larisamorales  staff   1350440 Jan 26  2021 HG002.sorted.dup.bai
-rw-r--r--@ 1 larisamorales  staff  17956229 Jan 26  2021 HG002.sorted.dup.bam
-rw-r--r--@ 1 larisamorales  staff   1350440 Jan 20  2021 HG002.sorted.dup.recal.bai
-rw-r--r--@ 1 larisamorales  staff  24793574 Jan 20  2021 HG002.sorted.dup.recal.bam
-rw-r--r--@ 1 larisamorales  staff     32357 Jan 20  2021 HG002.sorted.metric.insertSize.hist.pdf
```

Processes

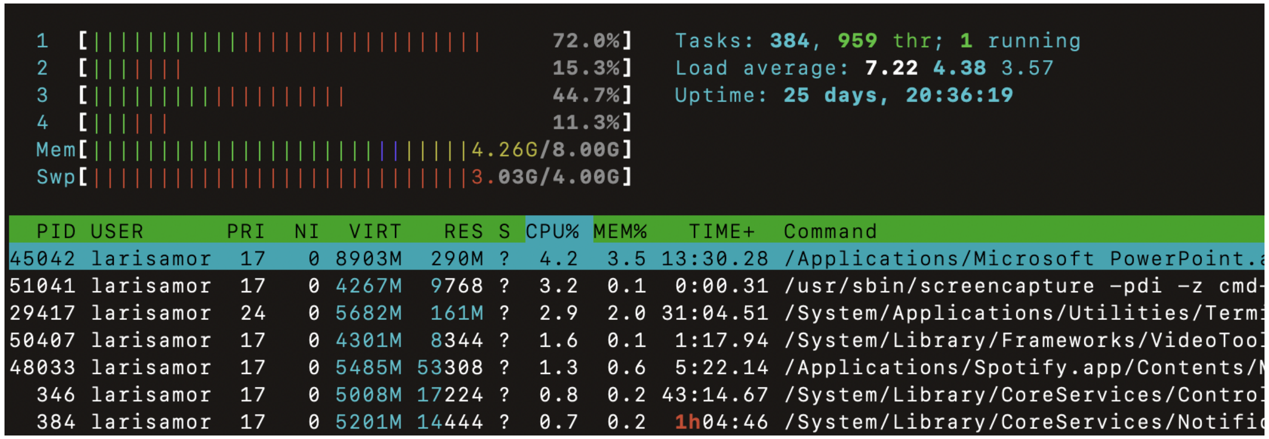


Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
zoom.us	1.7	52:25.01	26	262	0.0	0.00	47243	larisamorales
XprotectService	0.0	0.08	2	0	0.0	0.00	50628	root
XprotectService	0.0	0.06	2	0	0.0	0.00	50596	larisamorales
xartstorageremoted	0.0	2.62	2	0	0.0	0.00	235	root
WirelessRadioManagerd	0.0	0.13	2	1	0.0	0.00	50460	root
WindowServer	24.6	12:53:27.49	14	46	2.1	3:43:30.81	127	_windowserver
wifianalyticsd	0.0	2.60	2	0	0.0	0.00	293	root
Wi-Fi	0.0	5:28.75	4	0	0.0	0.00	369	larisamorales

System:	11.10%
User:	14.32%
Idle:	74.58%

CPU LOAD

Threads:	1,882
Processes:	384

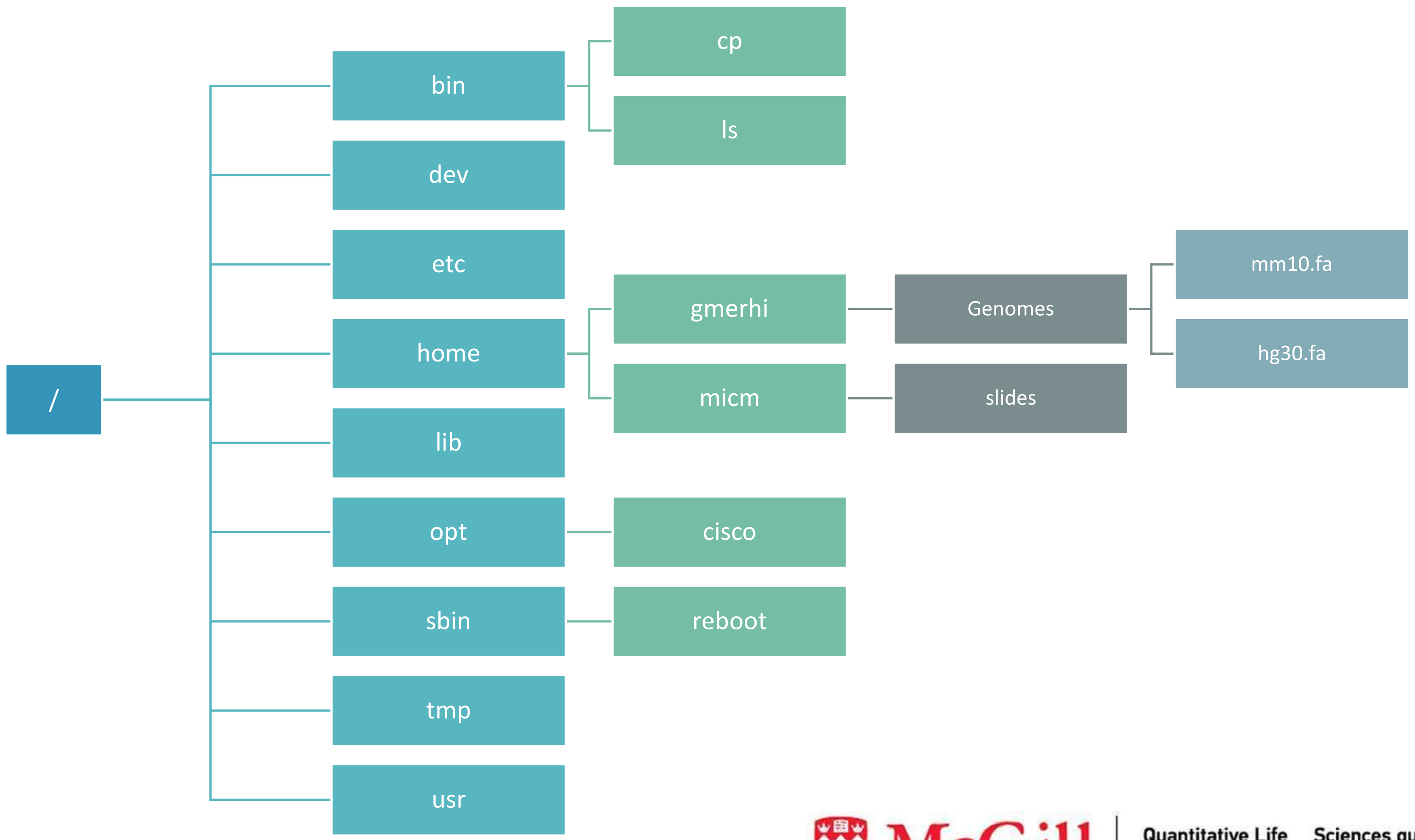


```
  1  [||||||||||||||||||||||||||||||        72.0%]   Tasks: 384, 959 thr; 1 running
  2  [|||||||                               15.3%]   Load average: 7.22 4.38 3.57
  3  [||||||||||||||||||||                  44.7%]   Uptime: 25 days, 20:36:19
  4  [||||||                                11.3%]
  Mem[||||||||||||||||||||||||||||||||4.26G/8.00G]
  Swp[||||||||||||||||||||||||||||||||3.03G/4.00G]

  PID USER       PRI  NI  VIRT   RES S CPU% MEM%   TIME+  Command
45042 larisamor   17   0 8903M  290M ?  4.2  3.5 13:30.28 /Applications/Microsoft PowerPoint.a
51041 larisamor   17   0 4267M  9768 ?  3.2  0.1  0:00.31 /usr/sbin/screencapture -pdi -z cmd-
29417 larisamor   24   0 5682M  161M ?  2.9  2.0 31:04.51 /System/Applications/Utilities/Termi
50407 larisamor   17   0 4301M  8344 ?  1.6  0.1  1:17.94 /System/Library/Frameworks/VideoTool
48033 larisamor   17   0 5485M 53308 ?  1.3  0.6  5:22.14 /Applications/Spotify.app/Contents/M
  346 larisamor   17   0 5008M 17224 ?  0.8  0.2 43:14.67 /System/Library/CoreServices/Control
  384 larisamor   17   0 5201M 14444 ?  0.7  0.2  1h04:46 /System/Library/CoreServices/Notific
```

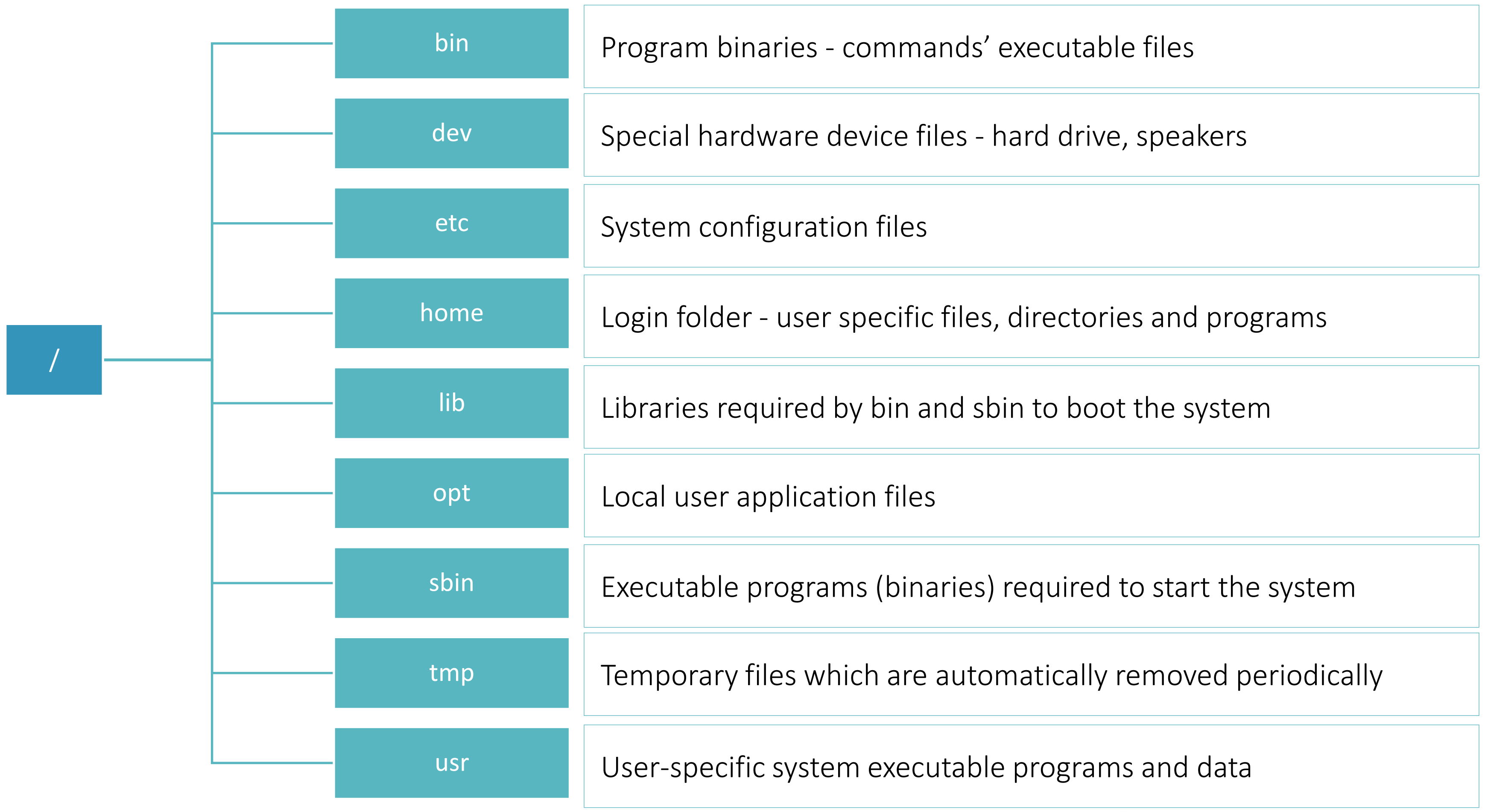
Quantitative Life Sciences
Sciences quantitatives du vivant

Everything is a **file** or a **process**

File system structure



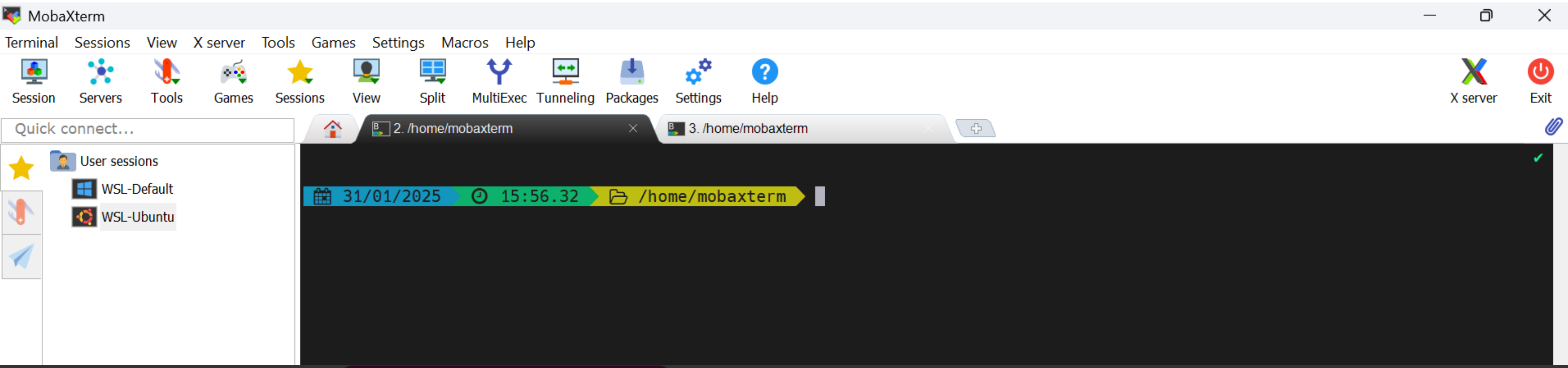
File system structure



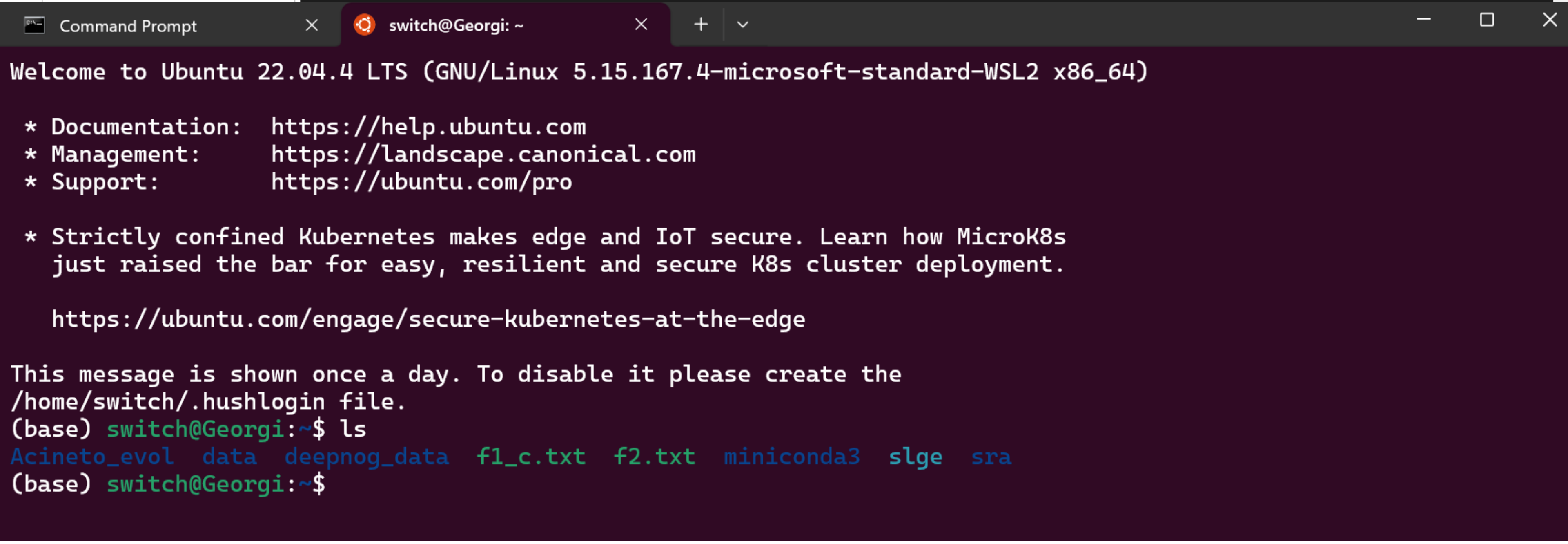
/		
	bin	Program binaries - commands’ executable files
	dev	Special hardware device files - hard drive, speakers
	etc	System configuration files
	home	Login folder - user specific files, directories and programs
	lib	Libraries required by bin and sbin to boot the system
	opt	Local user application files
	sbin	Executable programs (binaries) required to start the system
	tmp	Temporary files which are automatically removed periodically
	usr	User-specific system executable programs and data

McGill Quantitative Life Sciences Sciences quantitatives du vivant

The terminal


MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. /home/mobaxterm
3. /home/mobaxterm
User sessions
WSL-Default
WSL-Ubuntu

```
31/01/2025  15:56.32  /home/mobaxterm
```



```
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/switch/.hushlogin file.
(base) switch@Georgi:~$ ls
Acineto_evol  data  deepnog_data  f1_c.txt  f2.txt  miniconda3  slge  sra
(base) switch@Georgi:~$
```

Quantitative Life Sciences
Sciences quantitatives du vivant

Keyboard shortcuts

Ctrl+A	Ctrl+E	Ctrl+C	q
Go to the start of the line	Go to the end of the line	Stop the current process	Exist a child process (less, more, etc)

Unix commands

- Programs built in the shell that perform specific actions

Command | -[Options] | [Argument(s)]

```
$    ls    -lt    ~/Onedrive
```

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

Basic commands

top	man	history	clear
See active processes and the resources they're using	Shows the manual page of a command	List your previous commands	Clear your terminal window
% htop	% man ls % man cd % man htop	% history	% clear

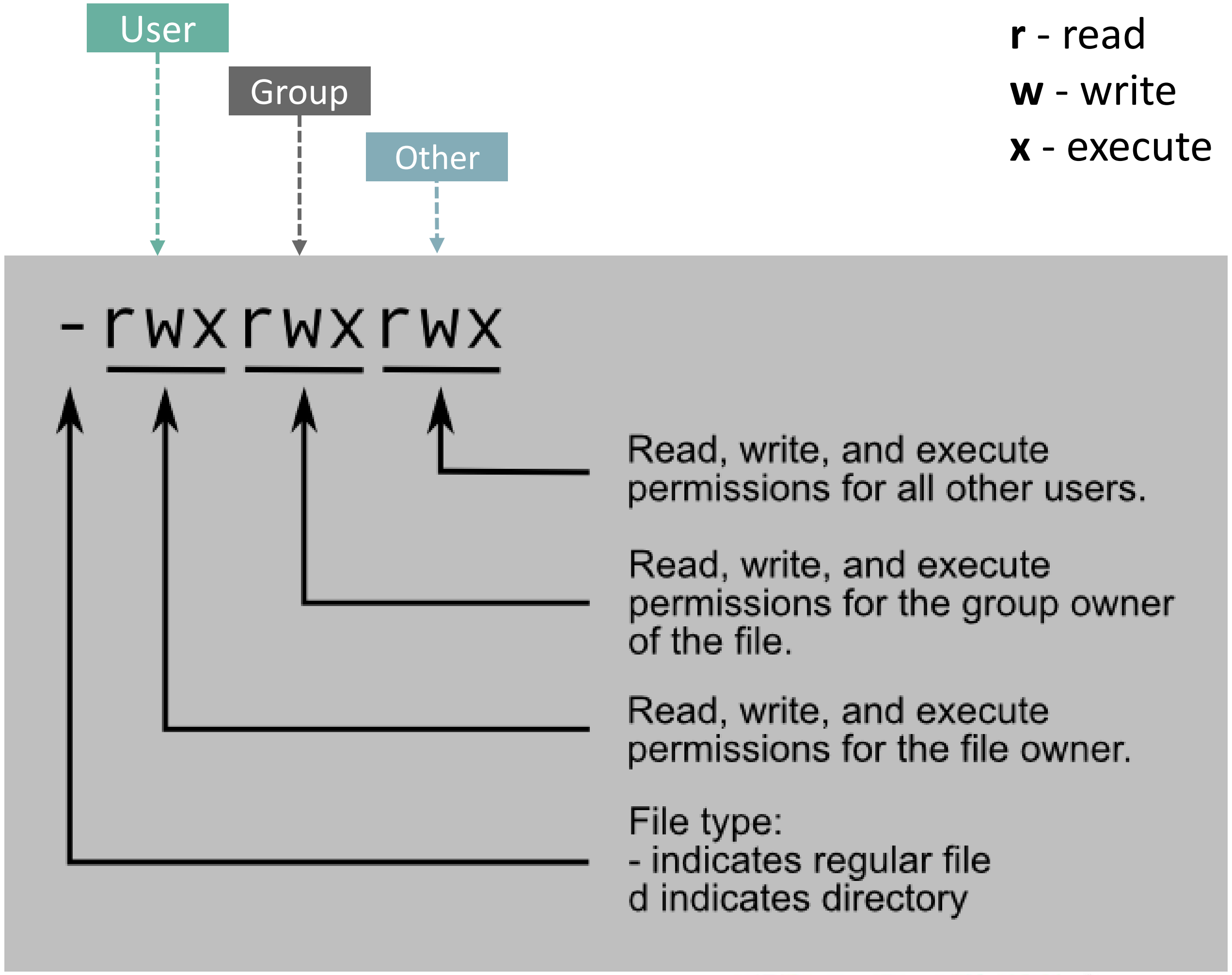
Basic commands

whoami	pwd	ls	cd
Displays current user id	Shows the current directory	Prints the current directory	Access a directory
% whoami	% pwd	% ls folder1	% cd folder1

Special characters

.	..	*	~
Current directory	Parent directory	Match any and all characters	Home directory
% ls .	% ls ..	% ls *	% ls ~

File permissions


User
Group
Other
-rwxrwxrwx
Read, write, and execute permissions for all other users.
Read, write, and execute permissions for the group owner of the file.
Read, write, and execute permissions for the file owner.
File type:
- indicates regular file
d indicates directory

r - read
w - write
x - execute

Adapted from https://linuxcommand.org/lc3_lts0090.php

File management commands

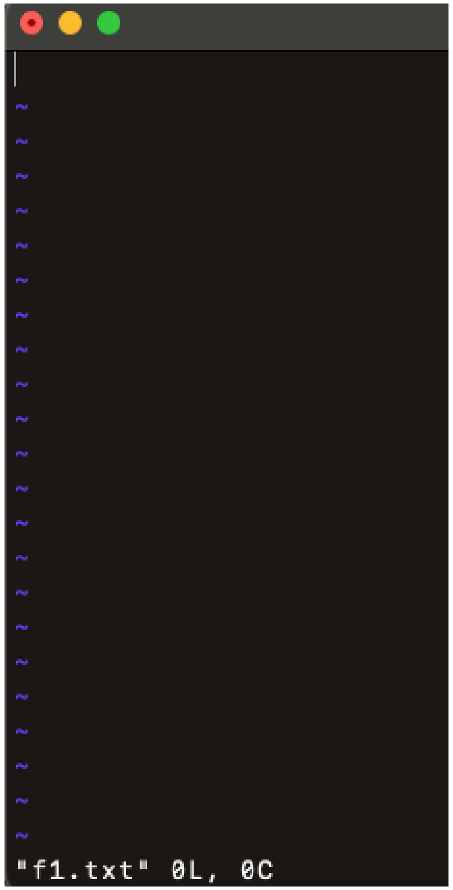
touch	mkdir	chmod	echo
Creates a new file	Creates a new directory	Change file permissions chmod **[ugo][+-][rwx]** u: user +: grant w: write g: group -: revoke r: read o: other x: execute	Prints something to the terminal
`% touch f1.txt`	`% mkdir ~/intro_unix`	`% chmod o-w f1.txt`	`% echo “hello world”`

File management commands

cp	mv	cat	zcat
Copy a file	Move a file or rename it	Print the contents of a file(s) to the terminal	Print the contents of a zipped file(s) to the terminal
`% cp f1.txt f1_copy.txt`	`% mv f1_copy.txt f1.txt`	`% cat f1.txt`	`% zcat f1.txt.gz`

How to edit files in the terminal

	nano	vi
	Recommended for plain text files	Recommended for code files/scripts
To open the file	`% nano f1.txt`	`% vi f1.txt`
To close the file	Press Ctrl+X Type Y Press Enter	Press Esc then hold Shift+zz
		Type :xq then press Enter



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"f1.txt" 0L, 0C
```

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

How to download a file and use it in the terminal

- % cd ~/intro_unix
- % mv ~/Downloads/cars.tsv data/ho1
- **scp [username]@[IP_address_of_the_server]:[path/of/files/on/server] [path/on/your/machine]**

File management commands

gzip	gunzip	tar	rm
Compress a file	Decompress a file	Bundle files with compression (optional)	Removes file(s)
% gzip cars.tsv	% gunzip *	% tar –cvzf cars.tgz f*	% rm cars.tsv

McGill Quantitative Life Sciences Sciences quantitatives du vivant

File management commands

head	tail	more	wc
Print the first N lines of a file	Print the last N lines of a file	View contents of a file	Count words, characters lines or bytes
% head -3 cars.tsv	% tail -3 cars.tsv	% more cars.tsv	% wc cars.tsv

Text processing commands

cut	sort	uniq	paste
Extract columns of file	Order elements	Get set of uniq elements	Paste two files column-wise
% cut –f1 cars.csv > col1.tsv	% sort cars.tsv > cars.sort.tsv	% uniq cars.sort.tsv	% paste col1.txt cars.tsv

Bonus skill

Create your own commands

```
% vi ~/.bash_profile
```

```
alias ll=“ls –l”
```

```
% source ~/.bash_profile
```

```
% ll
```

Hands on 1

1. Print the current directory and user
2. Create a directory named intro_unix in the home directory
3. Create a directory called data within intro_unix
4. Create a sub-directories ho1 ho2 and ho3
5. Create a directory called folder1 under data/ho1
6. Go into folder1 and create two files: f1.txt and .f2.txt
7. Write the numbers from 1 to 10 in f1.txt (one number per line)
8. Write the following sequence in .f2.txt (one letter per line)
   - **a a a b b b b c c c c c d d d d d d d**

1. List all the contents of the directory (including .f2.txt) – hint: look at the manual of the **ls** command
2. Change the name of .f2.txt to f2.txt
3. Change the permissions of f1.txt so that only the user can read and write the file
4. Write only the first 10 lines of f1.txt and all the lines in f2.txt to a new file called f3.txt

Bonus skill

1. Using the file cars.tsv :
    1. List the unique set of countries included in the dataset
    2. Find the number of unique car models

Module 2

- Pipes and redirects
- Variables
- Pattern matching
- Text processing

Pipes

- A way to connect the end of something with the start of something else
- They are specified with the control operator “ | ”

```
% cat cars.tsv | head -10
```

Redirect output

>	>>	<
Will send the output of the command to a NEW file	Will APPENND the output of the command to a file	Redirect output of one command as input to another command
% ls folder1 > files.txt	% ls folder1 >> files.txt	% head -1 <(cat files.txt)

Variables

- Used to store information
    - number, text, file name, etc
- The name can only contain:
    - Letters (a to z or A to Z)
    - Numbers (0 to 9) *not at the beginning
    - Underscore (“_”)
- We can have several types:
    - Environment variables
    - Shell variables
    - Global and local variables

Global variable

- Are variables that are accessible from any part of a script, within functions and sourced scripts.
- They do not disappear when the process is done but when the shell session is closed

```
% MY_NAME_LOCAL=I_am_groot

% echo $MY_NAME_LOCAL
```

Shell variables

- Environment variables required by the shell to function

`$USER`	`$HOME`	`$PATH`	`$PWD`
Stores the name of the current user	Has the path to the user's home	Contains all the directories where executable files are stored	Contains the path of the current directory

Environment variables

- Are available to any process in the shell

```
% export MY_NAME=Larisa

% echo $MY_NAME
```

Variable expansions

- Substituting the variable with its value
- It occurs before calling a command
- If the result contains spaces, the expansion should be quoted
- We can use operators to modify the content of the variable before passing it on to a command

Variable expansions

Allow us to modify the content of the variable before passing it to a command

- Extracting a part of the string

`${VAR:offset}`	Extracts the content starting from the offset
`${VAR%pattern}`	Extracts the content until it finds the pattern

Variable expansions

- Replacing substrings

${VAR/pattern/replace}	Changes **pattern** for **replace** the first time it appears
${VAR//pattern/replace}	Changes **pattern** for **replace** all the times it appears
${VAR%pattern}replace	Extracts content until **pattern** and writes **replace** after

Groupings

(…)

Subshell
Contains a list of commands. If preceded by $(…) then its expanded to the output of the commands

```
files=$(ls *xt)
```

{…}

Group commands and affect their parsing. When written as ${VAR} it is a **parameter expansion**

```
x=2
{x=4}
echo $x
x=2
(x=4)
echo $x
```

((…))

Arithmetic instructions
They operate at the level of numbers. The expansion is also $((…))

```
x=2
y=6
echo $(($x+$y))
```

Adapted from https://stackoverflow.com/questions/6270440/simple-logical-operators-in-bash

Example

Extract

```
% FILE_NAME=$(ls ../data/ho1 | tail -1)
% echo ${FILE_NAME:4}
% SUFFIX=${FILE_NAME:4}
```

Replace

```
% echo ${FILE_NAME%${SUFFIX}}.csv
% echo ${FILE_NAME%_*}.csv
```

Pattern matching

grep

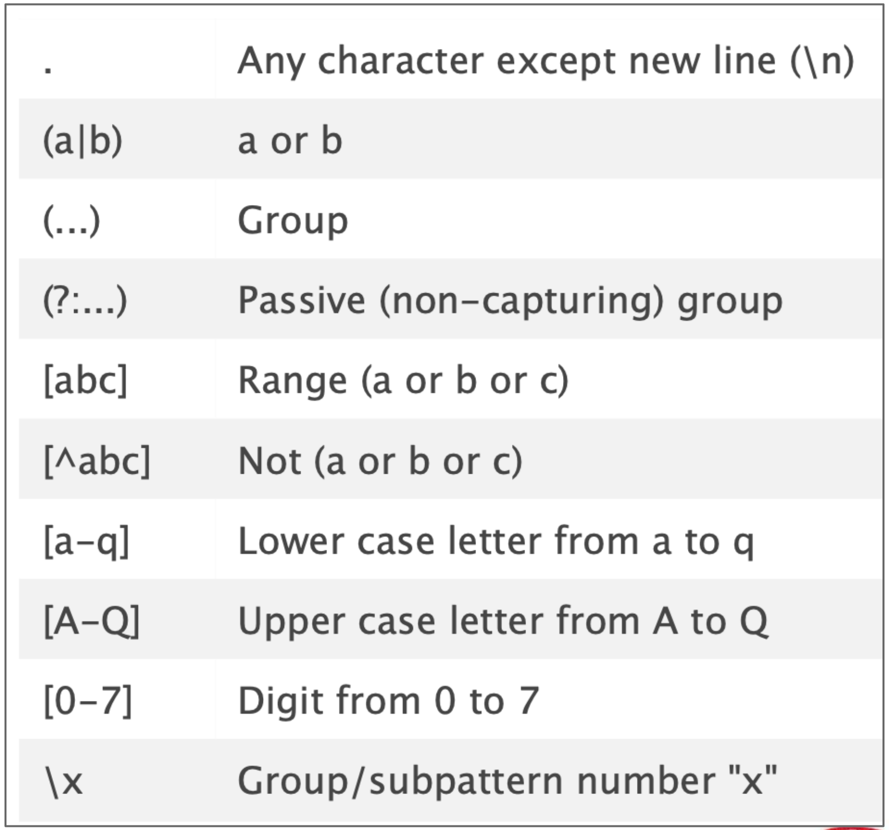
Search a pattern in a file

```
% grep Male
happiness.csv
```

```
Country Gender Mean N=
AT Male 7.3 471
AT Female 7.3 570
AT Both 7.3 1041
BE Male 7.8 468
BE Female 7.8 542
BE Both 7.8 1010
BG Male 5.8 416
BG Female 5.8 555
BG Both 5.8 971
```

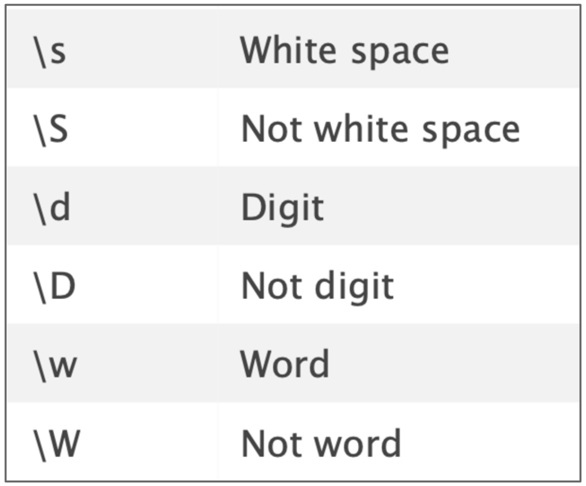
Regular expressions

Groups and ranges



.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"

Character classes



\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word

Adapted from https://cheatography.com/davechild/cheat-sheets/regular-expressions/

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

Regular expressions

Anchors

^	Start of string, or start of line in multi–line pattern
\A	Start of string
$	End of string, or end of line in multi–line pattern
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Special characters

^	[	.	$
{	*	(	\
+	)	|	?
<	>		

The escape character is usually \

Adapted from https://cheatography.com/davechild/cheat-sheets/regular-expressions/

Example

- All lines that start with a letter

```
% grep -E "^[A-Z]+" happiness.csv
```

- Lines that do not start with a letter

```
% grep -v -E "^[A-Z]+" happiness.csv
% grep  -E "^\W" happiness.csv
```

- Lines with a country from the list

```
% grep -f countries.txt happiness.complete.tsv
```

Bonus skills

awk

Scripting language that provides much more flexibility for text processing

Built-in functions

- length(string)
- tolower(string)
- toupper(string)
- match(string,pattern)

Built-in variables

- Entire line - $0
- Fields (specified by delimiter) - $1,$2,…

Example

- Print the 3rd column and then the 1st column

```
% awk '{print $3 "\t" $1}' happiness.complete.txt
```

- Print columns 1,3 and 2 if they contain the word “Female”

```
% awk '/Female/ {print $1 "\t" $3 "\t" $2}' happiness.complete.txt
```

- Count the number of characters in “Female” entries only

```
% awk -F "," '/Female/ { print length($0) "\t" $1 "\t" $2}'
happiness.complete.csv
```

sed

Command or streamline editor with multiple text processing functionalities

Built-in functions

- s/search/replace/ for pattern substitutions
  - /g – replace all occurrences
  - /1,/2,… - specifying which occurrence to replace
  - /I – Ignore case
  - /w – write to a file with /w filename
- -e to run multiple commands
  - sed -e 's/a/A/' -e 's/b/B/'

Examples

- Delete the first line

```
% ls -l | sed 1d
```

- Replace capital A for lowercase a

```
% sed 's/A/a/g' happiness.complete.txt | head
```

- Print the first line every 3 lines

```
% awk 'NR % 3 == 0' happiness.csv
```

Hands on 2.1

1. Go to the directory data/ho2 and list the files in it (move files there if missing)
2. Count many ”Female” and “Male” entries are listed in the file happinhowess.complete.tsv
3. Count how many unique countries are listed in the file happiness.complete.tsv

Hands on 2.2

1. Go to the directory data/ho2 and list the files in it
2. Using the file happiness.complete.tsv
   a. Write a new file in which all the countries are written in lowercase
   b. Replace “**Female**” for “**F**”, “Male for “**M**” and “**Both**” for **B** and look at the first 10 lines
   c. Write a new file with all the “**Both**” entries only for the countries included the file countries.txt

What will happen?

```
% export MY_NAME=LARISA
% MY_NAME_LOCAL=larisa
% vi my_name.sh
```

```
#!/bin/sh
echo "My name is ${MY_NAME}"
echo "My name is ${MY_NAME_LOCAL}"
```

```
% chmod u+x my_name.sh
% ./my_name.sh
```

Module 3

- Shell scripting
- Control structures
- HPC

McGill | Quantitative Life Sciences
Sciences quantitatives du vivant

For loop

```
for element in [list]
do
  [COMMANDS]
done
```

- This loop iterates over all the elements in the list
- Executes some command until the list is over

Numbers smaller than 10

```
for i in {1..10}
do
  echo number $i
done
```

Change file extension

```
for file in *tsv
do
  cp “$file” “${file%.tsv}.txt”
done
```

Adapted from https://linuxize.com/post/bash-while-loop/

Conditional statements

```
if [TEST]
then
  [COMMANDS]
fi
```

If the output of the test is ***True*** then the commands are executed. If it's false, nothing happens.

```
if [TEST]
then
  [COMMANDS1]
else
  [COMMANDS2]
fi
```

If the test is ***True*** then executed commands1 If it's ***False*** then executes commands2.

```
if [TEST1]
then
  [COMMANDS1]
elif [TEST2]
then
  [COMMANDS2]
else
  [COMMANDS2]
fi
```

If test1 is ***True*** it executes commands1. If it is ***False*** then evaluates test2. If test2 is ***True,*** executes commands2. If it is false, executes commands3

Operators for variables

-n VAR	*True* if the length of **VAR** is greater than zero
-z VAR	*True* if the length of **VAR** is zero
-v VAR	*True* if the variable **VAR** has been assigned a value
S1 == S2	*True* if the strings **S1** and **S2** are equal
S1 != S2	*True* if the strings **S1** and **S2** are not equal
S1 < S2	*True* if string **S1** sorts before **S2**
S1 > S2	*True* if string **S1** sorts after **S2**

Operators for arithmetic operations

INT1 –eq INT2	*True* if **INT1** is equal to **INT2**
INT1 –gt INT2	*True* if **INT1** is greater than **INT2**
INT1 –lt INT2	*True* if **INT1** is less than **INT2**
INT1 –ge INT2	*True* if **INT1** is equal or greater than **INT2**
INT1 –le INT2	*True* if **INT1** is equal or less than **INT2**

*Note: For arithmetic operations the statement need to be inside double square braces:

```
[[ INT1 -eq INT2 ]] in the case of integers
```

Operators for files

-f FILE	*True* if **FILE** exists and is a regular file
-d FILE	*True* if **FILE** exists and is a directory
-e FILE	*True* if **FILE** exists
-r FILE	*True* if **FILE** exists and is readable
-w FILE	*True* if **FILE** exists and is writeable
-s FILE	*True* if **FILE** exists and has a size greater than zero

Adapted from http://www.gnu.org/software/bash/manual/bash.html#Bash-Conditional-Expressions

Hands on 3.1 and solution

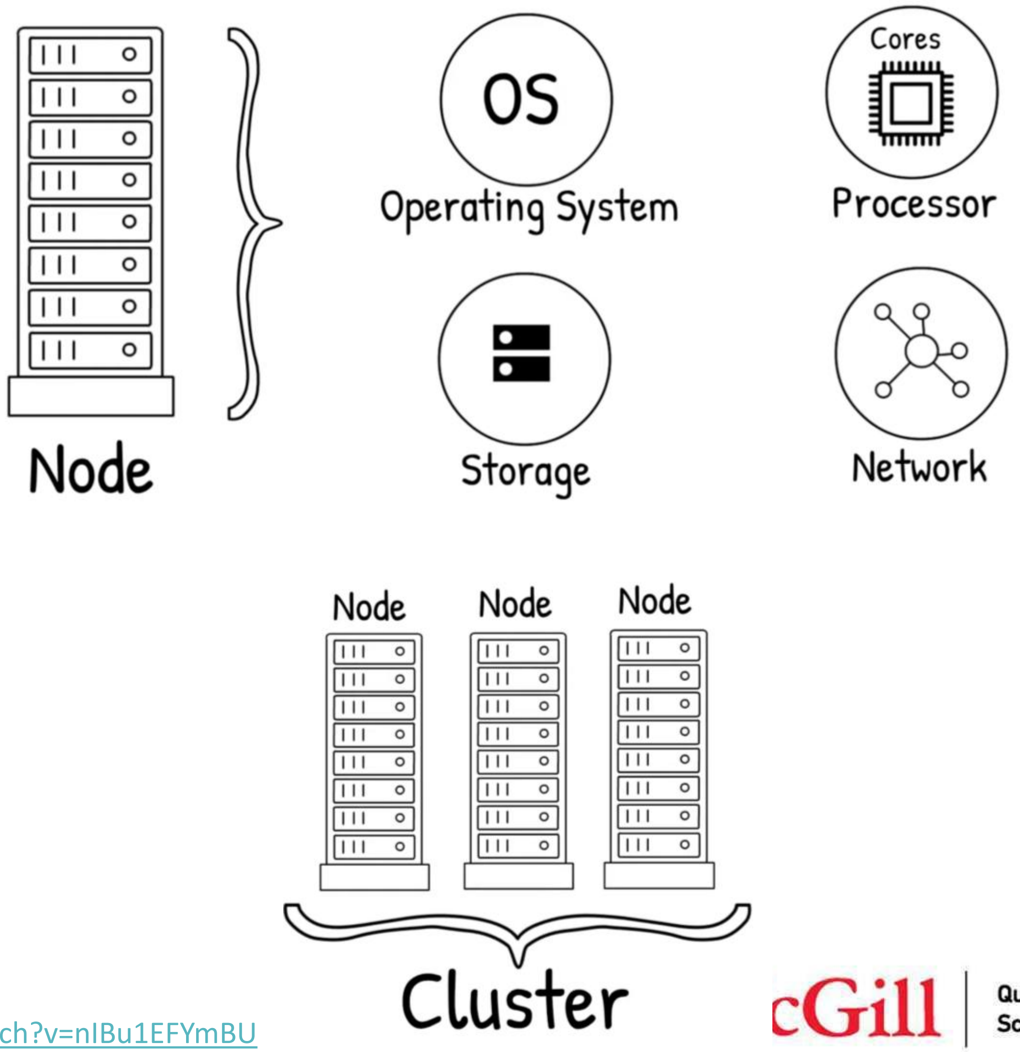
Counting and Analyzing Car Manufacturers

- You have a TSV file named cars.tsv in your ho1 current directory.
- The file contains data about cars, with at least the following columns: ID, Manufacturer, Model, Year.

Requirements:

- Write a Bash script to process the cars.tsv file and:
  - Count the total number of cars (excluding the header).
  - Count the number of cars for each manufacturer (e.g., Toyota, Honda, Ford).
  - Calculate the percentage of cars for each manufacturer.
  - Print the results in a readable format.

High performance computing

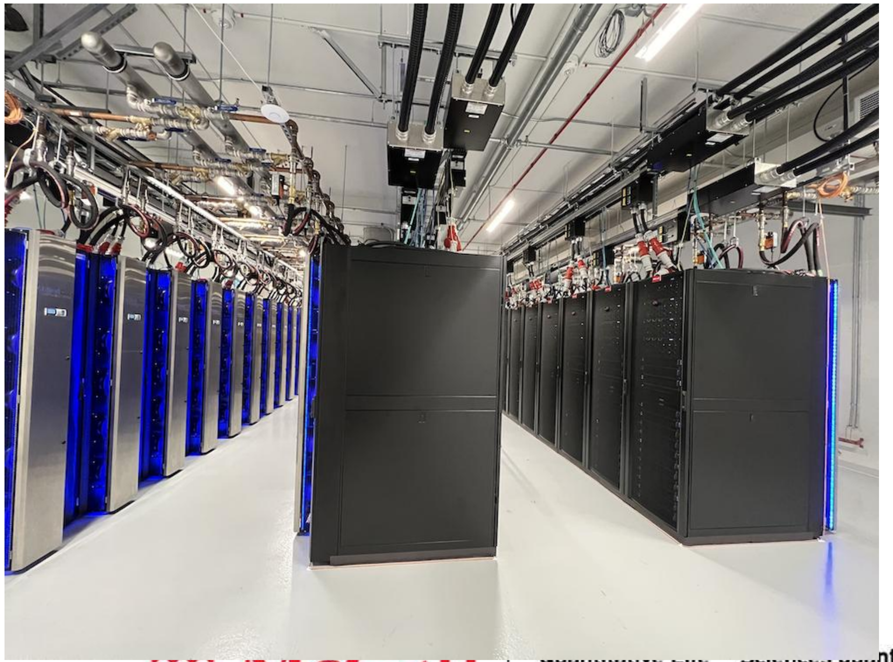


https://www.youtube.com/watch?v=nIBu1EFYmBU

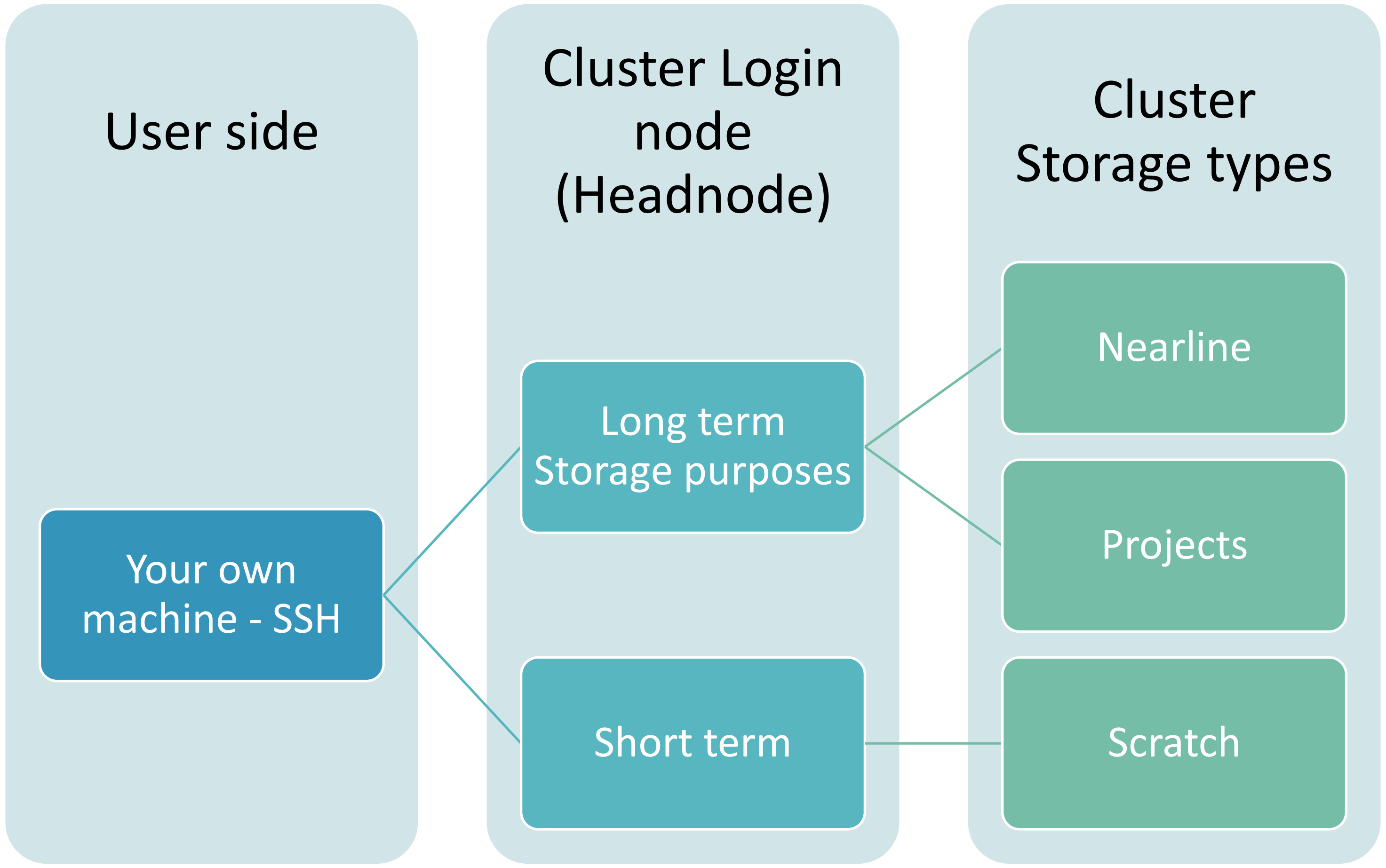
Digital Research Alliance of Canada (Compute Canada)

Digital Research Alliance of Canada | **Alliance de recherche numérique** du Canada

compute canada



Digital Research Alliance of Canada (Compute Canada)



Digital Research Alliance of Canada (Compute Canada)

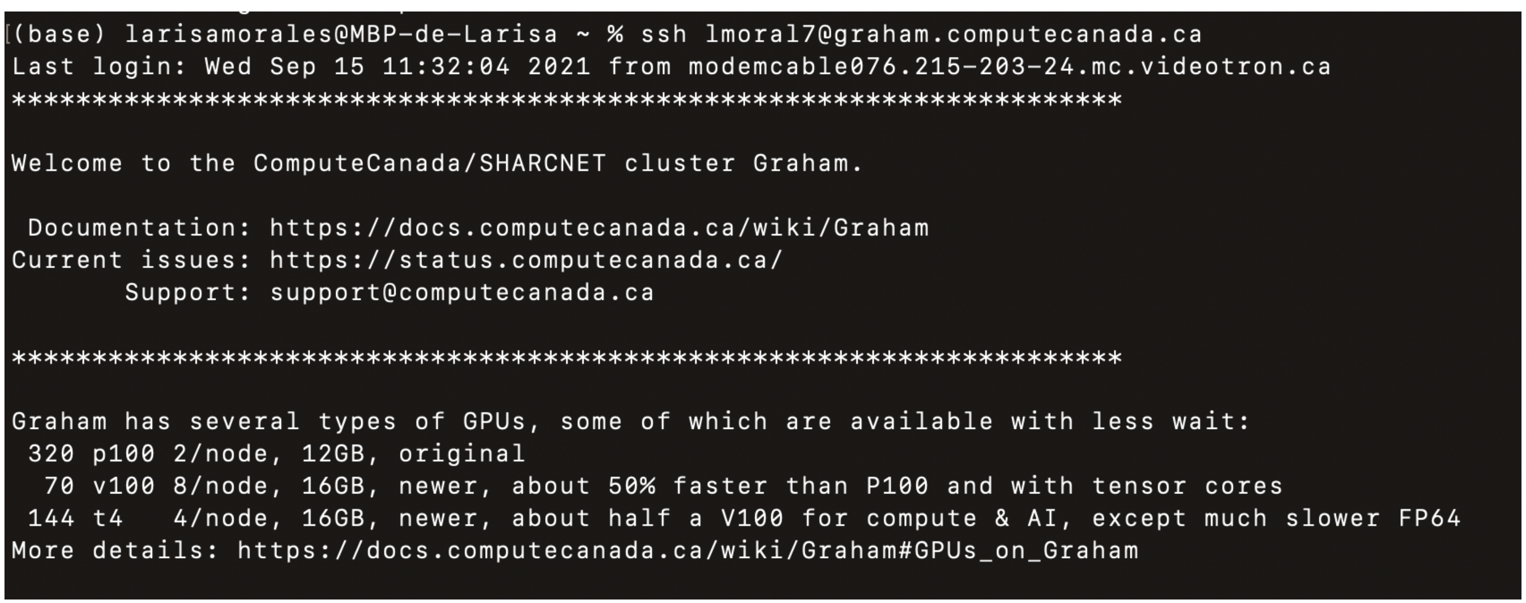
Feature	Scratch Storage	Nearline Storage	Projects Storage
Purpose	Temporary, high-performance storage	Long-term, infrequently accessed data	Persistent, project-related storage
Performance	High	Moderate to Low	Moderate to High
Capacity	Large	Large	Moderate to Large
Backup	No	Yes	Yes
Purge Policy	Regular purges	Data retained long-term	Retained throughout project duration
Accessibility	**Individual user**	**Individual user**	**Shared among project members**

Digital Research Alliance of Canada (Compute Canada)

- Overview of some important terms related to the cluster:
  - Compute Nodes → **one node = one computer**
  - Job Scheduling System → **SLURM**
  - SLURM job priority → **Fairshare**

Connect to a cluster

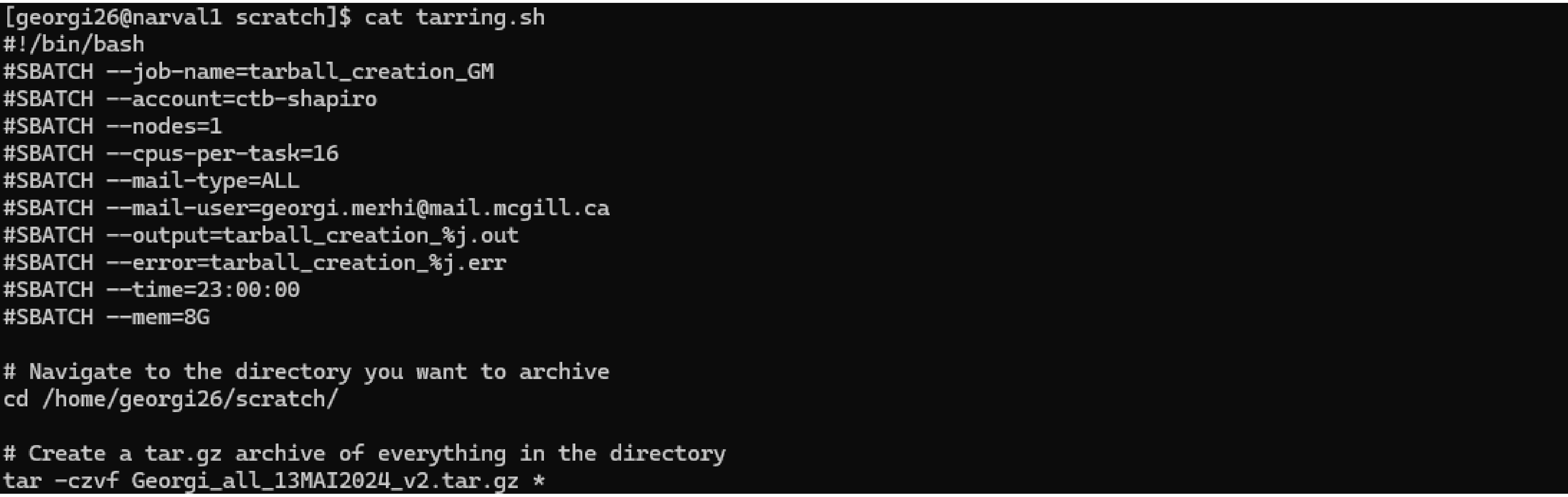
```
ssh user@cluster.computecanada.ca
```



```
(base) larisamorales@MBP-de-Larisa ~ % ssh lmoral7@graham.computecanada.ca
Last login: Wed Sep 15 11:32:04 2021 from modemcable076.215-203-24.mc.videotron.ca
*********************************************************************

Welcome to the ComputeCanada/SHARCNET cluster Graham.

 Documentation: https://docs.computecanada.ca/wiki/Graham
Current issues: https://status.computecanada.ca/
       Support: support@computecanada.ca

*********************************************************************

Graham has several types of GPUs, some of which are available with less wait:
 320 p100 2/node, 12GB, original
  70 v100 8/node, 16GB, newer, about 50% faster than P100 and with tensor cores
 144 t4   4/node, 16GB, newer, about half a V100 for compute & AI, except much slower FP64
More details: https://docs.computecanada.ca/wiki/Graham#GPUs_on_Graham
```

Jobs

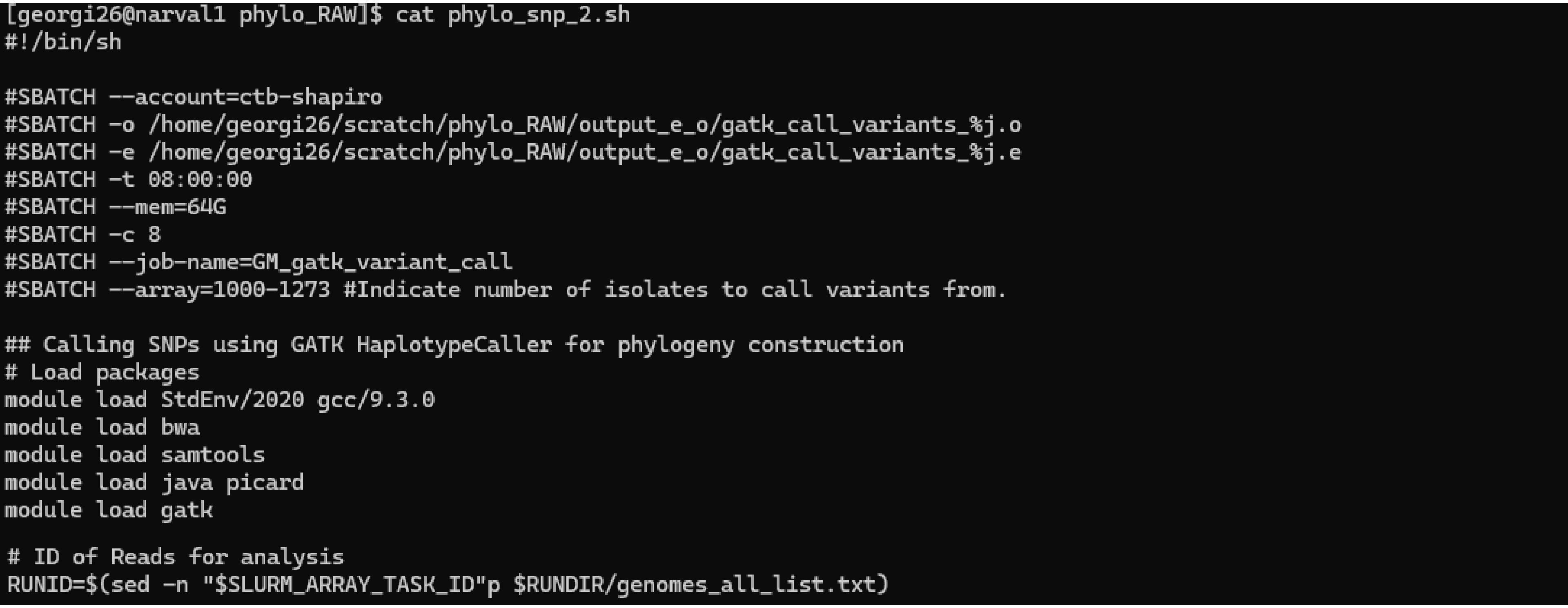
Requesting a computing allocation to execute your code



```
[georgi26@narval1 scratch]$ cat tarring.sh
#!/bin/bash
#SBATCH --job-name=tarball_creation_GM
#SBATCH --account=ctb-shapiro
#SBATCH --nodes=1
#SBATCH --cpus-per-task=16
#SBATCH --mail-type=ALL
#SBATCH --mail-user=georgi.merhi@mail.mcgill.ca
#SBATCH --output=tarball_creation_%j.out
#SBATCH --error=tarball_creation_%j.err
#SBATCH --time=23:00:00
#SBATCH --mem=8G

# Navigate to the directory you want to archive
cd /home/georgi26/scratch/

# Create a tar.gz archive of everything in the directory
tar -czvf Georgi_all_13MAI2024_v2.tar.gz *
```

Job arrays

A job script that internally behaves like a for loop and submits individual jobs



```
[georgi26@narval1 phylo_RAW]$ cat phylo_snp_2.sh
#!/bin/sh

#SBATCH --account=ctb-shapiro
#SBATCH -o /home/georgi26/scratch/phylo_RAW/output_e_o/gatk_call_variants_%j.o
#SBATCH -e /home/georgi26/scratch/phylo_RAW/output_e_o/gatk_call_variants_%j.e
#SBATCH -t 08:00:00
#SBATCH --mem=64G
#SBATCH -c 8
#SBATCH --job-name=GM_gatk_variant_call
#SBATCH --array=1000-1273 #Indicate number of isolates to call variants from.

## Calling SNPs using GATK HaplotypeCaller for phylogeny construction
# Load packages
module load StdEnv/2020 gcc/9.3.0
module load bwa
module load samtools
module load java picard
module load gatk

# ID of Reads for analysis
RUNID=$(sed -n "$SLURM_ARRAY_TASK_ID"p $RUNDIR/genomes_all_list.txt)
```

Jobs - Bonus

You can use the sbatch command with flags without adding parameter directives in a job script.

```
#!/bin/sh

module load iq-tree
iqtree2 -s ./core_gene_alignment_Acinetobacter_GM.aln -T 32 -m MFP -B 1000
echo done>run.done
```

Then run the command in the terminal as follows

```
sbatch -A ctb-shapiro --time 168:0:0 --tasks 32 --mem 800g run.sh
```

McGill | Quantitative Life Sciences Sciences quantitatives du vivant

Job submission/monitoring & Useful commands

sbatch job_script.slurm	Submit a job using a job script
squeue -u ${USER}	List all the processes of $USER
sq	Same as above but shorter
scancel [PID]	Cancel a process using its process ID (PID)
scancel -u ${USER}	Cancel all processes that belong to $USER
quota –s	Check your home directory usage and quota
quota --per_user	Check all users’ usage and quota

To summarize

- ✓ UNIX is an operating system with two parts: the kernel and the shell
- ✓ The shell (terminal) contains many built-in programs called commands
- ✓ Commands allow the user to perform operations on files
- ✓ HPC can help solve problems that go beyond the capacity of personal computers

Now you are ready to:

- Automate file management operations
- Extract relevant information from files without opening them
- Create pipelines that connect multiple tools
- Determine when a problem could be better solved with HPC

Thank you for attending!

1

Scan the QR code to confirm you attended today's workshop.

2

Fill out the feedback survey in the next 72h.

3

Get recognition for this workshop on your co-curricular record.

McGill | Quantitative Life Sciences | Sciences quantitatives du vivant

McGill initiative in
Computational Medicine

Datasets references

Hands on 1 – Cars dataset
https://perso.telecom-paristech.fr/eagan/class/igr204/datasets
Hands on 2 – Happiness surveys dataset
https://perso.telecom-paristech.fr/eagan/class/igr204/datasets

Resources
UNIX cheatsheet