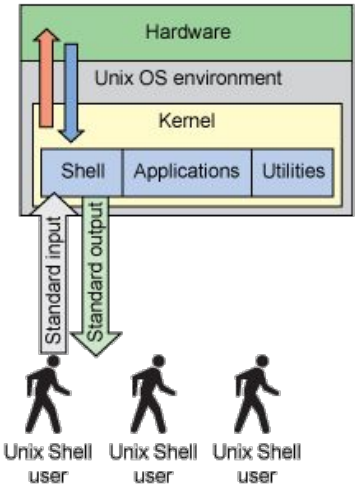
E4 + E5 Programming course 2025

Bash

Donna Maria Mattern, Lucas Cremer
based on slides from Aaron and input from Salvo (thanks!)
18.03.2025

What is Bash?

- Bash = Bourne-again shell
- most common shell amongst UNIX(-like) systems (eg. GNU, Linux...)
- user interface to the kernel (which manages the computer hardware) where user can send commands



https://askubuntu.com/questions/506510/what-is-the-difference-between-terminal-console-shell-and-command-line

- use tab to autocomplete commands for commodity
- default prompt: **[<username>@<node><dir>]$ <command>**

```
[dmattern@subra checkDiMuonSF]$ ls -lh
total 286K
drwxr-xr-x. 2 dmattern e4    8 Oct 14 16:16 basic_plotting
drwxr-xr-x. 2 dmattern e4    7 Oct 21 15:56 config
-rw-r--r--. 1 dmattern e4  18K Nov 26 10:10 fitDirectBalanceLargeRadiusJets.cxx
drwxr-xr-x. 2 dmattern e4    4 Nov 26 10:03 input
-rw-r--r--. 1 dmattern e4 8.3K Nov 26 10:04 MakeInsituResponseHistosLargeRadiusJets.C
drwxr-xr-x. 2 dmattern e4    4 Nov 26 09:37 noSF
drwxr-xr-x. 2 dmattern e4    6 Nov 26 09:41 results
drwxr-xr-x. 2 dmattern e4    6 Nov 26 10:05 results_pTll_100GeVcut
drwxr-xr-x. 2 dmattern e4    4 Nov 26 09:37 SFincluded
-rw-r--r--. 1 dmattern e4  12K Nov 26 09:57 slimCalibTreeLargeRadiusJets.cxx
drwxr-xr-x. 2 dmattern e4    7 Oct 21 15:56 utils
```

running a command in bash

Reminder: SSH

- ssh = secure shell, used to log in safely to infrastructure
- only accessible on E4 infrastructure: neptun (checkout IT onboarding session), then proxy jump to another workstation (put this in ssh config!)

- X11 forwarding to use GUI (Graphical User Interface) applications

logging into a workstation via ssh

```
[dmattern@subra ~]$ ssh -XY subra
dmattern@subra's password:
Last login: Wed Mar 12 12:29:39 2025 from 129.217.167.88

                    Welcome dmattern to subra

 Machine:                                    User:
  - Type:    working station                  - NFS home:  46.5 / 500.0 GB
  - OS:      AlmaLinux 9.4                    - Ceph home: 361 GB
  - N_users: 2 (terminal: 2, vscode: 1)
  - CPU:     0.4 / 12 threads used
  - Memory:  3.3 / 15.0 GB used
  - Uptime:  9 days

 INFO: This is a working station used for your daily work. Use it to develop, test and run your code. If you need
 more computational power, you can submit jobs to the condor cluster.

[dmattern@subra ~]$
```

ssh config

~/.ssh/config

```
Host neptun
  Hostname neptun.e4.physik.tu-dortmund.de
  User ldap-username
  IdentityFile /path/to/ssh/key
  ForwardX11Trusted yes
  ForwardX11 yes
  IdentitiesOnly yes
  ServerAliveInterval 15

Host proxy-condor
  User ldap-username
  Hostname condor
  IdentitiesOnly yes
  IdentityFile /path/to/ssh/key
  ForwardX11 yes
  ForwardX11Trusted yes
  ServerAliveInterval 15
  ProxyJump neptun
```

https://confluence.tu-dortmund.de/pages/viewpage.action?pageId=212337201

Bash commands 1: Navigation

Bash Cheat Sheet: https://github.com/RehanSaeed/Bash-Cheat-Sheet

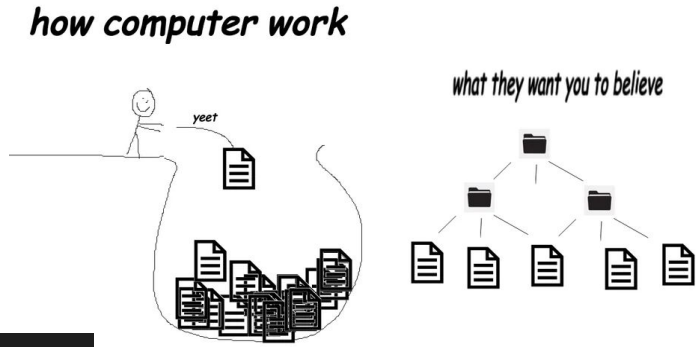
- current directory: **./**
- above directory: **../**
- home directory: **~/**
- previous directory:**-/**
- list directory contents: **ls <options> <dir>**
  - **-l** (list): show as basic list
  - **-a** (all): show all, including hidden files
  - **-h** (human-readable): shows size of files
  - Output:
    - filetype+rwx(owner, group, world) (r: read, w: write, x: execute)
    - number of files
    - owner, group, size
      (not reliable for directories! Use **du -sh** instead)
    - time of last change
- navigate between directories: **cd <path>**
- show current path: **pwd**

```
[dmattern@subra utils]$ ls -lah
total 148K
drwxr-xr-x.  2 dmattern e4    7 Oct 21 15:56 .
drwxr-xr-x. 10 dmattern e4   13 Nov 26 10:03 ..
-rw-r--r--.  1 dmattern e4  13K Oct 21 15:56 FitBalanceHelperFunctions.cxx
-rw-r--r--.  1 dmattern e4  14K Oct 14 16:16 JES_BalanceFitter.cxx
-rw-r--r--.  1 dmattern e4 4.6K Oct 14 16:16 JES_BalanceFitter.h
-rw-r--r--.  1 dmattern e4 8.9K Oct 14 16:16 JES_Smoother.C
-rw-r--r--.  1 dmattern e4 9.4K Oct 14 16:16 Util.h
[dmattern@subra utils]$
```

Bash commands 2: Creating files/directories

- create directory: **mkdir <name>**
  - **-p** (parent): creates parent directories if needed
- copy: **cp <path> <target>**
  - **-r** (recursive): needed eg. for directories
- move: **mv <path> <target>**
- create empty file: **touch <name>**
  (does not overwrite already existing files!)

Bash Cheat Sheet:
https://github.com/RehanSaeed/Bash-Cheat-Sheet




```
[dmattern@subra checkDiMuonSF]$ mkdir test
[dmattern@subra checkDiMuonSF]$ cd test
[dmattern@subra test]$ ls
[dmattern@subra test]$ cp -r ../input/ .
[dmattern@subra test]$ ls
input
[dmattern@subra test]$ ls input/
all.user.domatter.VjetsCalib_Rel22_01August23Run2_Zmumu_PowPy8_2016_ANALYSIS_nominal_slim_noTrigSF.root
[dmattern@subra test]$ touch test.txt
[dmattern@subra test]$ ls
input  test.txt
[dmattern@subra test]$
```

creating directories and files

Bash commands 3: Removing files/directories

Bash Cheat Sheet: https://github.com/RehanSaeed/Bash-Cheat-Sheet

- remove files, directories: **rm <path>**
  - **-i** (interactive)

  Caution! Recovery of data not always possible! Use -i option to be sure!

  - **-r** (recursive)

  Dangerous! Use for directories!

  - **-f** (force)

  Very dangerous! Removes write-protected files immediately!
  If a commands runs longer than you expect, take caution!

- remove empty directories: **rmdir <directory>**
  - Best practice: first remove files with **rm**, then remove empty files with **rmdir**.

Attention! ~ will overwrite everything before them, so don't do
rm /home/greatjulia/packages/badfiles/~/*

Bash commands 4: Wildcards

- useful to treat many similar named directories/files
  - ***** (any chain of characters)
  - **?** (any singular character)
- examples:
  - **rm *.pdf**: removes all files of type PDF in current directory
  - **cp Script*.py scripts/**: moves all python files starting with Script into script directory

listing files using wildcards

```
[dmattern@arrakis Run3_data_Sep_28_PRW_all_corr]$ ls NuisPar*.png
NuisPar_Btagging.png                NuisPar_Others.png
NuisPar_Electron_Uncertainties.png  NuisPar_PDF.png
NuisPar_Instrumental.png            NuisPar.png
NuisPar_Jet_Uncertainties.png       NuisPar_Theory.png
NuisPar_Muon_Uncertainties.png      NuisPar_Trigger.png
```

```
[dmattern@arrakis Histograms]$ ls emu_OS_?b_postFit.root
emu_OS_1b_postFit.root  emu_OS_2b_postFit.root
```

```
[dmattern@arrakis Histograms]$ ls [ee,mumu]*pre*.root
ee_OS_preFit.root       emu_OS_2b_preFit.root
emu_OS_1b_preFit.root   mumu_OS_preFit.root
```

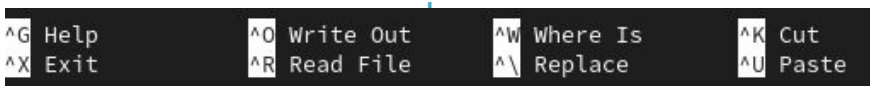
Advanced globbing options: https://en.wikipedia.org/wiki/Glob_(programming)

Wildcard	Description	Example	Matches	Does not match
`*`	matches any number of any characters including none	`Law*`	`Law`, `Laws`, or `Lawyer`	`GrokLaw`, `La`, or `aw`
		`*Law*`	`Law`, `GrokLaw`, or `Lawyer`.	`La`, or `aw`
`?`	matches any single character	`?at`	`Cat`, `cat`, `Bat` or `bat`	`at`
`[abc]`	matches one character given in the bracket	`[CB]at`	`Cat` or `Bat`	`cat`, `bat` or `CBat`
`[a-z]`	matches one character from the (locale-dependent) range given in the bracket	`Letter[0-9]`	`Letter0`, `Letter1`, `Letter2` up to `Letter9`	`Letters`, `Letter` or `Letter10`

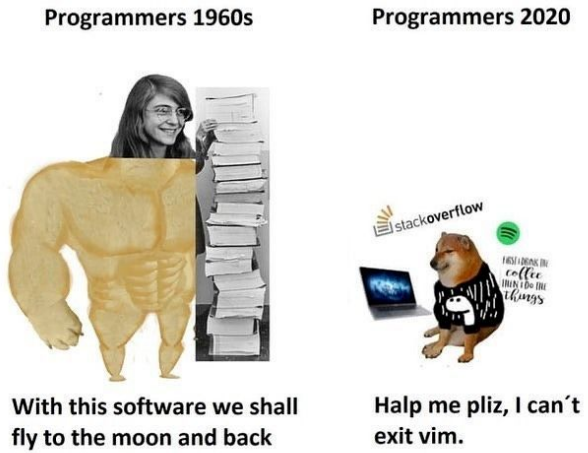
Text line editors

- nano: standard text editor in UNIX
  - can edit text as one is used to
  - **Ctrl+S** save current file
  - **Ctrl+X** close buffer, exit from nano
- vim: standard text editor in UNIX
  - different modes (normal/insert/visual/...)
  - **esc** enter normal mode
    - **:q** quit
    - **:q!** quit without saving
    - **:w** save
  - **esc+i** enter insert mode

Nano Cheat Sheet:
https://www.nano-editor.org/dist/latest/cheatsheet.ht



Vim Cheat Sheet:
https://www.cs.cmu.edu/~15131/f17/topics/vim/vim-cheatsheet.pdf



Bash commands 5: rsync

- copying files via ssh, requires working ssh config to be setup
  - **rsync <file/directory> <username>@<hostname>:/path**
  - works between local host and remote hosts or two remote hosts
  - you can not rsync from a remote machine onto your own laptop!
- alternatively use VS-code, mounts, …

Bash Cheat Sheet: https://github.com/RehanSaeed/Bash-Cheat-Sheet

```
[dmattern@arrakis ~]$ rsync note.txt dmattern@subra:/nfs/homes/dmattern
(dmattern@subra) Password:
```

using rsync to copy a file onto another workstation

Bash command help

- **help <command>**
  - key words and internal commands explained
- **man <command>**
  - manual
- **info <command>**
  - another documentation system like man
- **<command> --help**
  - another documentation system like man

Bash Cheat Sheet:
https://github.com/RehanSaeed/Bash-Cheat-Sheet

```
[dmattern@subra checkDiMuonSF]$ echo help
help
[dmattern@subra checkDiMuonSF]$ help echo
echo: echo [-neE] [arg ...]
    Write arguments to the standard output.

    Display the ARGs, separated by a single space character and followed by a
    newline, on the standard output.

    Options:
      -n        do not append a newline
      -e        enable interpretation of the following backslash escapes
      -E        explicitly suppress interpretation of backslash escapes
```

example usage of help and man to explain commands

```
[dmattern@arrakis backup]$ man --help
Usage: man [OPTION...] [SECTION] PAGE...

  -C, --config-file=FILE     use this user configuration file
  -d, --debug                emit debugging messages
  -D, --default              reset all options to their default values
      --warnings[=WARNINGS]  enable warnings from groff
```

Exercise 1: First steps in the terminal

- Try out the following things freely
  - navigate directories, including home and ceph
  - inspect and create directories and files
  - copy and move directories and files
  - remove directories and files
  - try-out wildcards
  - use rsync to copy to/from laptop/workstation

Feel free to ask us anytime!

What is .bashrc?

- your first bash script: **nano ~/.bashrc**
- user profile for your bash settings
- executed automatically when an interactive shell is started
- useful to define aliases
- Caution! condor-jobs can't access .bashrc and use the aliases!

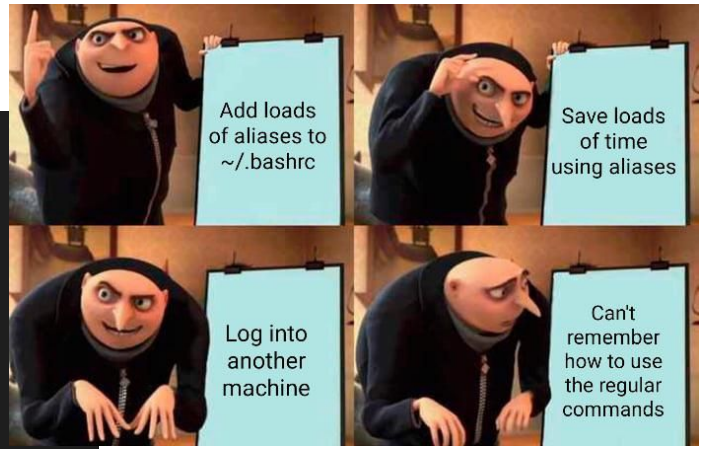
```
[dmattern@subra checkDiMuonSF]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
alias tt="export TERM=xterm-color"
export VSCODE_CUSTOM_LOCATION=/ceph/e4/users/dmattern/private/.vscode/

alias loadlcg="source /cvmfs/sft.cern.ch/lcg/views/LCG_103/x86_64-centos9-gcc12-opt/setup.sh"
alias sshlx="ssh -XY lxplus"
```



example aliases from my .bashrc

Bash scripts 1: Basics

- files ending in **.sh**
- contain list of commands that are executed in one shell via source script.sh
- practical for repetitive tasks, can grow quite complicated
- available keywords/commands
  - **if**, **else**, **break**, **while**, ...
  - **cd** , **ls**, **echo**, **pwd**, **touch**, ...
  - **functions**
  - **if**... **then**... **else**..., **case**, loops, ...
- Shebang: first line instructing to use bash, usually: **#!/bin/sh**
  - you can also use **#!/<path/to/python>** to use python code in the script

```
[dmattern@subra checkDiMuonSF]$ cat script.sh
#!/bin/sh
echo "Hello world"
[dmattern@subra checkDiMuonSF]$ source script.sh
Hello world
```

executing a simple bash script

Bash scripts 2: Advanced options

- variables
  - **=** definition
  - **$** access
  - Attention! Spaces and tabs change the code!
- arrays
  - **array=(value1 ... valueN)** initialize
  - **${array[0]}** access singular element
  - **${array[*]}** access all element
- addition, subtraction, multiplication, division possible for integers
  - for floats/doubles: use external programs, eg. python
- loop options: **if**, **else** as usual
  - **fi**: closes if statement
  - **then**: in combination with if
  - **do**: in combination with loops
  - **done**: ends a loop

spaces in bash scripts matter

```
[dmattern@subra checkDiMuonSF]$ cat script1.sh
#!/bin/sh
var1=100
var2=500
echo $var1 $var2
[dmattern@subra checkDiMuonSF]$ cat script2.sh
#!/bin/sh
var1 = 100
var2 = 500
echo $var1 $var2
[dmattern@subra checkDiMuonSF]$ source script1.sh
100 500
[dmattern@subra checkDiMuonSF]$ source script2.sh
bash: var1: command not found...
bash: var2: command not found...
100 500
```

```
[dmattern@arrakis backup]$ if [ 1 -eq 1 ]; then echo "1=1"; fi
1=1
```

```
for i in 1 2 3
do
  echo "$i"
done
```

examples for usage of loops and if cases

Bash scripts 3: Arguments and execution

- arguments can be passed to bash script
  - could make them more adaptive eg. to pass paths
  - **$$** values of arguments
  - **$#** number of arguments

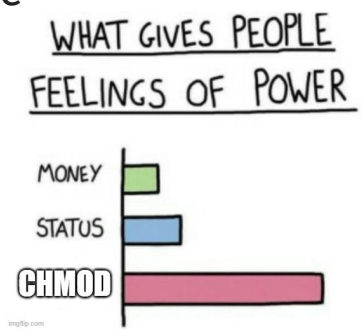
```
[dmattern@subra checkDiMuonSF]$ cat script3.sh
#!/bin/sh

echo $0
echo $1
echo "to"
echo $2
[dmattern@subra checkDiMuonSF]$ source script3.sh Hi you
-bash
Hi
to
you
```

shell script with arguments

Other ways of executing scripts:

- **./script.sh**
  - executes script in a new shell andpastes output to previous shell -> only .bashrc loaded, previously defined variables not available
  - needs execution rights on script (run **chmod u+x script.sh**)
  - shebang is required to make clear which shell should be used
- **bash script.sh**
  - executes script in new bash



Bash scripts: Overview

- powerful tool, practical for different tasks, but slightly cryptic
- current status:
  - many scripts already out there that you can adapt for your needs
  - … or use chat GPT to write them
- sometimes more useful to write python scripts

example from Aaron

```
#!/bin/bash

# Load python 3.6+ as it is needed -> load_sw is an alias, jsut load some proper cvmfs software package
load_sw

# Define the paths where the corresponding fit output files can be found
path_CB_Z="/eos/user/v/vandergr/Run3_2024_Validation_New/Merged/out/"
path_CB_Jpsi="/eos/user/v/vandergr/Run3_2024_Validation_New/Merged/out/"

#Define the paths where the plots shall be written to
out_path_CB="/eos/user/v/vandergr/WebEOS/Validation_2024_April/"
#out_path="/eos/user/v/vandergr/WebEOS/Rel22_Validation_2024/"

cd /afs/cern.ch/user/v/vandergr/private/MomentumValidation/source/MomentumValidation/scripts/

python MMC_full_plotting_refactored.py -r Z -t CB -f Eta -i $path_CB_Z -o $out_path_CB --directCB True --extension .png --which_run Run-3 --which_period e
python MMC_full_plotting_refactored.py -r Jpsi -t CB -f Eta -i $path_CB_Jpsi -o $out_path_CB --directCB True --extension .png --which_run Run-3 --which_period e
python MMC_full_plotting_refactored.py -r Z -t CB -f Pt -i $path_CB_Z -o $out_path_CB --directCB True --extension .png --which_run Run-3 --which_period e
python MMC_full_plotting_refactored.py -r Jpsi -t CB -f Pt -i $path_CB_Jpsi -o $out_path_CB --directCB True --extension .png --which_run Run-3 --which_period e


# Spread index file for WebEOS
cd /eos/user/v/vandergr/WebEOS
source Spread_index_file.sh
cd ~
```

Bash commands 6: Inspecting files

```
[dmattern@subra checkDiMuonSF]$ grep -r "open" basic_plotting/
basic_plotting/makeBasicHists.cxx:  auto lumiMC   = openSettingsFile("config/MCandLumi.data");
basic_plotting/makeBasicHists.cxx:  auto fileData = openSettingsFile("config/Files.data");
basic_plotting/makeBasicHists.cxx:  auto histData = openSettingsFile("config/Histos.data");
basic_plotting/makeSlimPlots.cxx:  auto histData = openSettingsFile("config/Histos.data");
[dmattern@subra checkDiMuonSF]$
```

using grep to find specific string in all files in a directory

- **cat <file>**
  - prints total file
- **head <file>**
  - prints first 10 lines
  - -n: specify how many lines
- **tail <file>**
  - prints last 10 lines
  - -n: specify how many lines
- **grep <string> <file>**
  - find lines in files with specific string
  - -i: case insensitive
  - -R: recursive (search multiple files)
  - -A, -B: give A (b) previous(subsequent) lines
- **du -sh <file or directory>**
  - show size of file/directory
  - Useful command! Sometime you might reach your quota limit!

What is w?

- shows all shells that are currently running on this machine
  - **TTY**: shell name
  - **IDLE**: time since last input in the shell
  - **JCPU**: CPU-time for shell (not precise)
  - **PCU**: which process is running (not precise)
- before rebooting your machine, check if someone else is working there!
  Or if it randomly gets very slow...

running w to find all running shells on arrakis

```
[dmattern@arrakis ~]$ w
 13:19:02 up 31 days,  2:05,  4 users,  load average: 1.18, 1.38, 1.33
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
ndueser  pts/25    13:07    4.00s  0.11s  0.11s -bash
dmattern pts/33    12:13    0.00s  0.08s  0.01s w
dkoskows pts/26    13:07    4:37   0.03s  0.03s -bash
voliveir pts/35    13:17   30.00s  0.02s  0.02s -bash
[dmattern@arrakis ~]$
```

Permanent ssh connection: tmux

tmux Cheat Sheet:
https://tmuxcheatsheet.com

- ssh connection based on data exchange
  -> issues when Internet cuts, laptop lid is closed, …
  - broken pipe errors common, running processes are killed with ill-defined behavior
- **tmux new -s <mysession>** start a new tmux session called mysession
- **tmux kill-session -t <mysession>** kill session called mysession
- **tmux a -t <mysession>** attach session called mysession
- **CTRL+b d** detach current session
- **CTRL+b w** session and window preview
- **CTRL+b c** create new window
- …

Permanent ssh connection: screen (alternative)

screen Cheat Sheet:
https://gist.github.com/jctosta/af918e1618682638aa82

screen -S <mysession>	start a new screen session called mysession
screen -ls	list active screen sessions
screen -x	attach a running session
screen -r <mysession>	attach running session called mysession
screen -d <mysession>	detach a running session
CTRL+a d	detach from current session
CTRL+a :	exit screen
...	

tip: add this to your .screenrc to enable scrolling in screen sessions

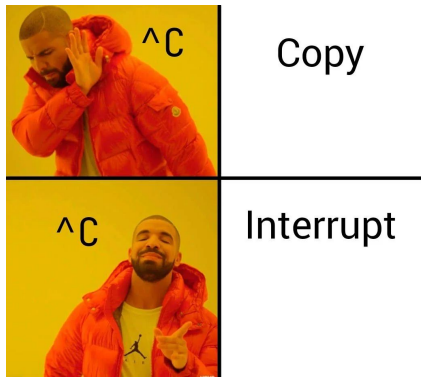
```
[dmattern@arrakis ~]$ cat ~/.screenrc
# Enable mouse scrolling and scroll bar history scrolling
termcapinfo xterm* ti@:te@
```

Terminal hotkeys

- **CTRL+C**
  - copy
- **CTRL+V**
  - paste
- **CTRL+d**
  - close terminal
- **CTRL+c**
  - terminate process (SIGINT)
- **CTRL+z**
  - suspend process (SIGTSTP)
  - could restart process at a later time

```
PRE JVT: 21.302702 small R subjet p
PRE JVT: 11.709183 small R subjet p
PRE JVT: 9.674524 small R subjet pT
PRE JVT: 18.882111 small R subjet p
^C
 *** Break *** keyboard interrupt
```

GNU termination signals Cheat Sheet:
https://www.gnu.org/software/libc/manual/html_node/Termination-Signals.html



Overview

- Nobody knows all commands, you learn piece-by-piece and you can look things up
- useful to look at cheat sheets
- What you should take away from today
  - shell scripts are very useful for repetitive workflows, but a bit tricky
  - many are already available online or can be written with help from Chat GPT, but for this you need to know what is possible!

Exercise 2: Bash scripting

- Try out the following things freely
  - try writing a bash script to create directory structures
  - look at the cheat sheet commands, try them out and understand them
  - use to ssh-shells to test out these commands: w, htop
  - run a tmux or a screen session, where you log in and out

Feel free to ask us anytime!