Interactive Use / History

Tab completion	press Tab (1-3 times). Many things complete. Common: paths
ctrl-r key combo	Search backwards in history
up arrow	last command from history
`history`	show your command history

Finding Documentation

`man command (e.g. man cp)`	Manual pages for commands. Search for words with /
`info command`	info pages. Often more verbose than man pages
`command --help (e.g cp –help)`	help flag for most gnu commands
`help`	list bash internal commands
`help <command>`	help for bash internal commands
`https://mywiki.wooledge.org/`	Bash Internet Resource (Bash FAQ, Bash Guide)

File system paths

. and ..	current directory and parent directory
/	root directory
~ or $HOME	expanded by the shell to the path of your home dir
cd ~/parent/child	Change current directory to "child"
cd ..	Change directory from “child” to parent directory "parent"

Listing directory contents, Disk Space

`ls`	„list“ - List the files in a directory.
`ls -la`	-a: show hidden files -l: long format
`du`	„disk usage“, how much space is in use (including subfolders)
`du -hs dir1`	-h = "human readable" -s = "summary"
`df -h`	„disk free“ - show total free space for mounted volumes

File transfer

scp	Secure copy ($ scp <sourcefile> <username>@<host>:<targetfile>) with -r: recursively
sftp	Secure file transfer program ($ sftp <username>@<host>:<targetdir>)

Shared file access - change permissions for different groups and users

`chgrp mygroup file`	Makes file belong to the group mygroup
`chmod u=rw,g=rw,o=r file.txt`	user (u): read + write group (g): read + write other (o): read only
`chmod -R a+r dir`	Change permission recursively on „dir“, „all“ get read permission

Moving, renaming, and copying files

`cp file1 newfile`	Copy a file
`mv file1 newname`	Move or rename a file. cp and mv are used in the same way
`mv file1 ~/AAA/`	Move file1 into sub-directory AAA in your home directory
`rm file1 [file2...]`	Remove or delete a file. with -r: recursively (**Careful**!)
`mkdir dir1 [dir2...]`	Create directories. With -p: create all needed directories
`rmdir dir1 [dir2...]`	Remove an empty directory

Viewing and editing files

`less filename`	Progressively dump a file to the screen: /word = search for word; SPACE = page down Page-Up or U: up ; q=quit
`nano filename`	Edit a file using the “nano” editor
`vi filename`	Edit file using “vi” or “vim” (see section below)
`head -n 8 filename`	Show the first 8 lines of a file
`tail filename`	Show the last few lines of a file
`tail -f filename`	-f: "follow" - keep showing lines as the file grows forever

Environment variables

`DIRROOT=/usr/local/dir`	Defines the variable DIRROOT with the value /usr/local/dir
`export DIRROOT=/usr/local/dir`	Exports the variable to a child process
`cd $DIRROOT`	Changes your present working directory to the value of DIRROOT
`echo $DIRROOT`	Prints out the value of DIRROOT, or /usr/local/dir
`env and printenv`	Print all available environment variables

Standard environment variables:

$PATH	All directories with executables
$USER	User name
$HOME	User's home folder (/home/user_name)
$TMPDIR	Special variable for temporary folder (/scratch/user_name)

Searching

`grep “string” file_name`	Prints all the lines in a file that contain the string
`grep -il “string” file_name`	Print filename(l) if file contains “string”, **i**gnore-case(i)
`find /mnt -name xyz.txt`	Finds file “xyz.txt” recursively from directory “/mnt”
`find . -type f`	Finds all files that are regular files under current directory (.)
`find /path/to/somedir -type f -name 'some*name*' -exec grep -il 'regexp-pattern' '{}' +`	Finds a file from point “somedir”, the name of file consists of “some*name”, and file has line with “regexp-pattern”

Redirection / Pipes

cmd > out.txt; grep string filename > out.txt	stdout → file; stderr still on terminal. File gets overwritten(!)
cmd 2> out.txt	stderr → file; you still see non-error output on the terminal
cmd &> out.txt	stdout AND stderr → file
cmd 1>&2	stdout → stderr; often used as cmd > file 2>&1
cmd 2>/dev/null	stderr → NULL (ignore errors)
`grep string filename >> existfile`	Appends the output of the grep command to the end of 'existfile'
`ls -l | less`	Output of "ls -l" is sent with “|” (“piped”) to the command "less".
`du -sc * | sort -n | tail`	"du -sc *" lists sizes for all files and directories, "sort -n" orders the output from smallest to largest size, "tail" displays last few lines

Archives

`tar cvf archive.tar file1 …`	Create (c) a tar archive as a file “archive.tar” containing file1…
`tar tvf archive.tar`	List (t) the contents of “archive.tar“
`tar xvf archive.tar`	Extract (x) from the archive file
`tar cvfz archive.tar.gz dir`	Create gzip compressed(z) tar

Process management

`top and htop`	Interactive list of processes (htop Extended version of “top”)
`ps aux`	List of the current processes
`ps aux | grep $USER`	use grep to see only your own processes
`ps aux | grep firefox`	See all processes of “firefox”
`pidof »process_name«  or`	Find out process id (PID)
`ps aux | grep »process_name«`	
`kill »pid_of_process«`	Kill process (per PID)

Loops

`for key in a b c; do echo $key;done`	repeat something with a, b and c in $key
`for file in out* ; do ls -la $file; done`	do something with all files in the directory starting with out

Shell globbing with Wildcards –- the shell parses wildcards and variables, not the command!

`? (question mark)`	Any single character
`* (asterisk)`	Any number of characters (e.g. `find file*.txt`)
`[ ] (square brackets)`	Specifies a character range e.g. [A-Z] is any capital letter
`\ (backslash)`	Protect a subsequent special character

Regular Expressions

grep, sed, awk, vim,...	programs that use regular expressions
`.`	Any character
`[ ] (square brackets)`	character range e.g. [ab] is a or b, [A-Z] any capital letter
`*`	Any number (incl. 0) of the preceding character
`^ $`	^ at the start: Begin of the line $ at the end: end of the line
grep ^[ab]*$	find lines that ar empty or only contain the letters a or b

vi / vim

`i`	„insert“ - enter edit mode
`ESC Key`	exit edit mode / enter command mode
`:wq`	:w (write) and :q (quit) – save and quit
`:q!`	force quit without saving
`/`	search for a word / regular expression
`dd`	delete a line
`p (or  P)`	insert the last deleted thing below (or above)
long cheat sheet:	https://vim.rtorr.com/

awk

	/bla/: execute code in { } on lines with „bla“
awk '/bla/{print $3;sum+=$4}	print $3: print the 3rd column of the file
END{print sum}'	END{} execute code in braces at the end of file
awk '$4>100{print}'	print lines with value in 4th column bigger than 100
awk -F#	separate columns on #, not spaces. Uses Regex

Linux Bash Shell Cheat Sheet

for HPC Users

K.Siegmund