**uniq** - report/omit repeated adjacent lines

-c = --count = prefix lines by the num of occurrences

-d = print only duplicate lines

**sort** - sort lines of text files

-d = alphanumeric characters (default)

-n = numeric-sort

-r = reverse

-f = ignore-case

-u =remove duplicates, output just unique lines

-t "x"= field delimiter "x" (default = white space)

-k M[,N] =sort field key=part from col M and EOL or col N

*sort -t"," -k1,2 -k3n,3 = on 1st and 2nd then on 3rd num*

*sort -t"," -k1,1 -u = remove duplicates based on 1st field*

**cut** - slice lines

-d "x"= field delimiter "x" (default = TAB)

-f =select only these fields *(1,2,4-6)*

--output-delimiter=STRING (default= same as input)

**paste** - Concatenate horizontally; Merge lines of files

= with NO arguments on 1 file = cat command

-s = join all lines in a file

-d "xy"= delimiters "xy" (default = TAB)

- - - = num of "-" equals num of columns in output

*paste file1 file2 vs <file1 <file2 vs - - <file1 <file2*

*paste <(seq 10) <(cat text.txt)*

**tr** *-option SET1 SET2* = translate/delete chars

-s = squeeze-repeated chars from SET1

-d= delete chars in SET1, do not translate

*tr -d "a-z" tr -d "[:digit:]"*

-c = keep just the characters set with -d option

**grep** “STRING" [files] = print lines matching pattern

-v = Invert match; select non-matching lines.

-i = case unsensitive

-n = Prefix each line with its line number

-c = print count of matching lines for each input file

-w = only lines containing whole word matches

- [A/B/C] +N = print lines after/before/around match

-H = Print the file name for each match.

-E = enable regular expression

-o = show just the pattern matched

-b = show byte offset of the starting point of match

**sed** - line oriented **s**tream **ed**itor

-i = edit files in place

-n = no printing (default:print every line)

s = substitute+delimiter+in+out *sed 's/day/night/'*

/g = global -> all occurrences of the pattern

/I = case insensitive *sed 's/this/THAT/gI'*

p = print (with “-n” = print modified lines) *sed -n '2,4p'*

d = delete line seq 5 | sed '/3/d’ **vs** *seq 5 | sed -n '2,4d'*

! = reverse the restriction *sed -i '1~3!d' file.txt*

*sed -n 's/pattern/&/p' <file = grep pattern*

**Regular expression** = pattern that describes set of strings.

. = any character except newline

\w \d \s = word, digit, whitespace

\W \D \S = not word, digit, whitespace

[abc] = any of a, b, or c

[^abc] = not a, b, or c

[a-g] = character between a & g

^ = matches the beginning of line

$ = matches the end of a line.

Repetition operators:

? = item is optional and matched at most once.

\* = zero or more ocurrences.

+ = one or more ocurrences.

{n} = n number of ocurrences.

{n,} = min number of ocurrences.

{,m} = max number of ocurrences.

{n,m} = min-max number of ocurrences

ab|cd = match ab or cd

Escaped special characters

\. \\* \\ escaped special characters

\t \n \r tab, linefeed, carriage return

*Test Playground:* [*https://regexr.com/*](https://regexr.com/)

**Character classes:** (there are more classes)

[:alnum:] all letters and digits

[:alpha:] all letters

[:blank:] white spaces

[:digit:] all digits

[:lower:] all lower case letters

[:upper:] all upper case letters

**zip** out.zip file1.in file2.in= compress several files

-r = directory

**unzip** = list, test, extract

-p = print content

-c = extract to stdout (print name of each file)

-p = extract to stdout (without file namea)

*unzip -p text\_files.zip one\_file\_from\_zip|less*

-l = list files

**zipinfo** = list detail information

**zcat, zless, zgrep** = cat, less, grep over zip

**gzip** = compress **one** file to file.gz

-d = decompress

-f = force = overwrite existing files

-l = list compression info of gz file

-k = keep input file (default = compress in-place)

*gzip file1 file2 file3 -> produces 3 gz files*

**gunzip** = decompress

**zcat, zless, zgrep** = cat, less, grep over gz

**bzip2** = compress **one** file to file.bz2

*Hadoop read, manipulate and slice in blocks (64/128MB)*

-d = decompress

-f = force = overwrite existing files

-k = keep input file (default = compress in-place)

--best /--fast = compression methods

**bunzip2 =** decompress

**bzcat, bzless, bzgrep=** cat, less, grep over bz/bz2

**tar** = archiving files utility

-c = create

-r = add

-x = extract

-t = list/view

-f [FILE]= file archive (needs to be followed by name)

-v = verbose

-z = zip

-j = bzip2

-C -destination = extract to destination directory

*tar -czvf opentravel.gz.tar \*.csv*

*mkdir optd; tar -xzvf ./opentravel.gz.tar -C optd*

**Job handling (*per shell)***

**CTRL+C** = kill a job in foreground

**&** = run the command as background job

**CTRL+Z** = suspend the current foreground job

**bg** = move suspended job to background.

**jobs** = lists the active jobs

-n = show new jobs that changed status from last call

-r = display running jobs

-s = display stop jobs

**fg** = bring susp/bkground job to foreground.

= no arguments = most recent job

%x = bring to fg the job with ID=x *(ID from jobs)*

**kill** = kill the process by ID or PID

%x= kill bg/susp job from same shell

PID=kill the process by its process ID *(tab to get PID)*

**xkill** = kill a process by selecting a window **pkill** = Kill the process by name *(use tab)*

**pgrep** = look up process based on name

**top** = display Linux processes

**htop** = interactive process viewer

**ps** =snapshot of current processes *(use with grep)*

-e = select every process

-f = full listing

-U = select process by real user

**PPID =**parent PID; it started PID (use with zombie processes)

**csvlook** = render a file as a fixed-width table.

-d = delimiter

**csvstat** = descriptive statistics for each column

-H =csv file has no header row

-l = show line numbers

**csvcut** = like "cut" cmd; output delimiter ","

-c = column

-n = display column names and indices

**csvgrep** = like "grep" cmd; output delimiter ","

-m = pattern

-i = invert the result

**csvsort** = like "sort" cmd; output delimiter ","

-r = reverse

-n = display column names and indices

**csvformat** = convert to custom output format

-D = output delimiter

**csvstack** = stack up rows from multiple files

**csvjoin** = execute a SQL-like join to merge files

**csvsql** - generate SQL table create statement

-i = select SQL dialect *(sqlite,mysql, postgresql …)*

**if-then-elif-fi** *conditional expr in [ ] with space around*

*a=10 ; b=20*

*if [ $a == $b ] ; then echo "a is equal to b"*

*elif [ $a -gt $b ] ; then echo "a is greater than b"*

*elif [ $a -lt $b ] ; then echo "a is less than b"*

*else echo "None of the condition met"*

*fi*

**for-do-done** *seq of characters separated by spaces*

*for var in word1 word2 ... wordN or for var in $(seq 1 10)*

*do*

*echo $var*

*done*

**while-do-done until-do-done**

*while [ "$a" -lt 10 ] ; do until [ ! $a -lt 10 ] ; do*

*echo $a echo $a*

*a=`expr $a + 1` a=`expr $a + 1`*

*done done*



**date** = print or set the system date and time

**bc** = calculator *(echo 1+2 | bc )*

**expr** = evaluates the given expression

**column** = put list into columns

**split** = split a file into pieces

**diff** = compare files line by line

**md5sum** = compute and check MD5 message digest

**#!** = hash(**she**)+exclamation mark (**bang**) *(#!/usr/bin/bash)*

create new symlink (fails if symlink exists):

ln -s /path/to/file /path/to/symlink

create or update a symlink:

ln -sf /path/to/file /path/to/symlink

Try: ls -lH and compare the results with ls -l

**Distribution**=made by taking Linux core + some tools

**cat /etc/os-release**

**Kernel**=core app; allocates resources & talks to HW

**uname -r (-a)**

Latest: **dnf list kernel**

Install**: dnf install kernel-devel --best**

**sudo dnf update kernel** 🡪 reboot

*sudo dnf –y update = update all*

**Shell**=app that interprets the commands

Current: **ps $$** or **echo $0**

Default: **echo “$SHELL”**

List: **chsh -l** or  **cat /etc/shells**

Change: **chsh -s *shell\_name*** 🡪 log out

**Terminal**=app where we type the commands

**ps -p$PPID** (term app creates shell, so it is the parent of the shell)

**echo $TERM** (term type; tells apps how to interact with term)

Prompt = system symbol of cmd line (***#,$,%,:***)

Continuation prompt: **>** (continuation of previous line)

Breaking cmd in various lines: **\** or **|**

Separating 2 commands at one line: **;** or **&&**

Autocomplete opens with tab + ↑↓ + enter

Depends on the context (*cd +tab vs cp +tab*)

**CTRL+shift+n** = open shell in new window

**CTRL+shift+t** = open shell in new tab

**CTRL + l** = clear screen

**CTRL + r** = history block search

**CTRL+D** = terminate the shell

**ALT+b/f** = move backward/forward word by word

**CTRL + u** = cut/erase the whole line

**CTRL + k** = cut/erase line right from the cursor

**CTRL + w** = cut/erase word left

**ALT + d** = cut/erase word right

**CTRL + y** = paste (1st buff)

**CTRL+SHIFT+c =** copy highlight text (2nd buff)

**CTRL+SHIFT+v =** paste 2nd buff; after usage=1st buff

**ALT + c** = capitalise first letter of the word

**ALT + u** = uppercase the rest of the word

**ALT + l** =lowercase rest of the word

**who** - show who is logged on

**whoami** - print userid

**pwd** - print current directory (= **echo $PWD**)

**man** *cmd* = manual (*cmd -h* or *cmd --help*)

**type** *cmd* - type of a cmd tool

**-a** : all occurrences of cmd name

**which** *cmd* - which binary are you executing?

*which cmd* vs *sudo which python*

**whereis** *cmd* - location of the binary/source/man files **history** – last 15 commands

**-100** = last 100 commands

**-i** = include all information

***echo $HISTFILE*** *🡪 ~/.history*

*!+number\_hist\_line (!!=repeat last cmd -> sudo !!)*

**echo** - send argument to stdout

**-n** = doesn’t add new line character

**cat** - send content of file to stdout

**-n** = add number to all output lines

**head** - show 10 first lines of file

**-n K** = first K lines instead of 10

**-c K** = first K bytes

**-n/c -K** = all but the last K lines/bytes

**tail** - show last 10 lines of file

**-n K** = the last K lines instead of 10

**-c K** = last K bytes

**-n/c +K** = starting with K lines/bytes

**-f** = output appended data as the file grows;

**/** = root directory

**./** = current directory

**../** = upper (parent) directory

**~** = user home directory

**.***name* = hidden dirs/files start with dot!

*name***~** = backup files

**\** = escape character (split cmd line, special char)

**$** = preceding variable name (“\$” to print $)

**$0** = name of the running process.

**$(cmd)** =cmd substitution

**$((...))** = arithmetic expansion operator

# sizeof

**|** = pipe🡪use output of cmd 1 as input to cmd 2

**0<** = stdin **1>** = stdout **2>** = stderr **&>=** stdout&err

stderr by default is going to the console as stdout

*cat < file > file\_content 2> error\_content*

**>** = stdout redirection 🡪 overwriting the output file

**>> =** stdout redirection 🡪 appending to output file

**<** = take stdin from file (wc < file, <file wc, wc file, cat file | wc)

**2>&1** = redirect (add) errors to stdout

**/dev/null** = null device; discard all data & ret success

**cd** - navigate between dirs

= with NO arguments takes us to ~

**-** = toggle between the last two dirs.

**mkdir** - make a directory

**-**p = make parent directories as needed

**touch** - creates empty file/updates access & modif time

**cp** - copy a file/ directory

**mv** - move/rename files/directory

*cp/mv -options source destination*

**-r** : recursive mode used for directories

**-i** : interactive confirm file overwriting

**-v** : verbose see copy progress

**-p**: preserve file permission/attributes

**rm** - eliminate files

**-f** : force, never prompt

**chmod** - change file read/write/execute permissions

**ugo** = user/group/other (a=all)

**rwx**= read/write/execute

u(**rwx** )/g(**rwx** )/o(**rwx**)->9 binary->3 decimal->ex:737

ex: u+r+w,g-w,o+wrx (NO space in parameters)

**ls** – print the contents of the current dir

**- 1** = 1 output per line

**- s** = size

**- l** = long = all information

**- a** = all -> hidden directories/files start with dot!

**- H** = follow symbolic links

**- R** = list subdirectories recursively

**- d =** do not enter inside directories

**- S** = sort by file size

**- t** = sort by modification time, newest first

**- X** = sort alphabetically by entry extension

**- r** = reverse order while sorting

Pattern matching @command line

\* = match all files and subdirectories (show subdir content)

\*x = restrict to files and subdirectories starting with x

\*x\* = restrict to files and subdirectories containing with x

\*x = restrict to files and subdirectories ending with x The \* = any number of unknown characters,

? = only one unknown character

^ = negation (*\*(^/)=any pattern not having ”/” inside*)

*If restriction result is empty NO filter is used*

List just directories **: ls -d \*(/)** **; ls -d \*/** ; **echo \*/**

List just files: **ls -a \*(^/)**

List hidden dir/files = **ls -ld .\***

Get files/dirs with abs path: **ls -d -1 $PWD/\***

For entering 2nd level: **ls -d -1 $PWD/\*/\***

**’ ’** - single quotes=do not touch this text

**" "** - double=perform shell variable expansion

**` `** - evaluate & replace=cmd substitution (**`≠‘**)

== **$(cmd)** BUT **≠** **$cmd**

**wc** - print line, word, and byte counts

**-c** = print the byte counts

**-m** = print the character counts

**-l** = print the newline counts

**-w** = print the word counts

**seq** - print sequence of numbers (*start step stop*)

**-f** = format ( *-f %5.1f, -f%3.1e, -f “Line: %g”* )

**-s** = delimiter (default = \n)

**less** - interactively show content of a file

**-N** = show line number

**-S** = truncate lines wider than window

Use this while reading:

**G / g =** go to end/beginning of file

**q** = quit

**/** = forward Search (**?** = backward search)

**^**pattern : pattern @ beginning of line

pattern**$** : pattern @ end of line

**n** – next match (**N** = previous match)

{ } 🡪 parameter expansion

{a,b{1..3},c} = a b1 b2 b3 c

mv log{,.OLD} = mv log log.OLD

echo {00..8..2} = 00 02 04 06 08 echo {D..T..4}

🡪 variable identification

VAR=AB; echo $VAR12; echo ${VAR}12

🡪 Text replacement, after find & xargs = {}

🡪 Block of code = { cmd1; cmd2; . . . cmdN; }

( ) 🡪 evaluate & replace

🡪 array creation = array=(1 2 3)

🡪 subshell creation = pwd; (cd /; pwd); pwd

(( )) 🡪 arithmetic operations:

((a = 42)) ((a++)) echo $((a + b + (14 \* c)))

for ((i=0; i<10; i++))

[ ] 🡪 test commands (*man test*)

[ "$foo" -lt 3 ] or [[ $bar =~ ^123 ]]

🡪 range or character class

ba[rz], foo[[:alnum:]], qu[[=u=]]x

🡪 part of an array assignment

f=(3 4); f[42]=bar; echo $f,$f[2],$f[3],$f[42]

[[ ]] 🡪 Extended test construct builtin

**find** - search for files *[path] [conditions]*

**-type** + f=file, d=directory

**-name =** find by name (**-iname** = case insensitive)

*find . -type f -name "text\_file\*"*

**-maxdepth/mindepth** = max/min dir levels *(Level 1=./)*

**-perm p =** with permissions p *(p is integer ex: 757)*

**-not** = **!** = invert the match

**-size +/-n**= file larger/smaller than n (**-empty**)

**-mmin** **N** = files modified within N minutes

**-mtime N** : files modified within N days

**-newermt** YYYY-MM-dd = modified on or after date

**-exec** cmd = execute command on every found file

**-ok** cmd = prompt before executing on a file

*find \*.txt -exec ls {} \; -exec* ***sh -c*** *"head {} | tr A B” \;*

*All occurrences of {} are replaced by the filename.*

**dir=(\*)** = store dir content in array

**du -a --max-depth=1** = disk usage

**df .** = amount of available disk space for current dir

**tree -f -L 2** = contents of dirs in a tree-like format.

export GIT\_EDITOR=vim -> kwrite