**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 by process ID (*shell PID= echo $$, 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)*