Bash-Oneliner

A collection of handy Bash One-Liners and terminal tricks for data processing and Linux system maintenance.

View on GitHub

Bash-Oneliner

I am glad that you are here! I was working on bioinformatics a few years ago and was amazed by those single-word bash commands which are much faster than my dull scripts, time saved through learning command-line shortcuts and scripting. Recent years I am working on cloud computing and I keep recording those useful commands here. Not all of them is oneliner, but i put effort on making them brief and swift. I am mainly using Ubuntu, Amazon Linux, RedHat, Linux Mint, Mac and CentOS, sorry if the commands don't work on your system.

This blog will focus on simple bash commands for parsing data and Linux system maintenance that i acquired from work and LPIC exam. I apologize that there are no detailed citation for all the commands, but they are probably from dear Google and Stackoverflow.

English and bash are not my first language, please correct me anytime, thank you. If you know other cool commands, please teach me!

Here's a more stylish version of Bash-Oneliner~

Handy Bash one-liners

Terminal Tricks

Using Ctrl keys

```
Ctrl + n : same as Down arrow.
Ctrl + p : same as Up arrow.
Ctrl + r : begins a backward search through command history.(keep pressing Ctrl
Ctrl + s : to stop output to terminal.
Ctrl + q : to resume output to terminal after Ctrl + s.
Ctrl + a : move to the beginning of line.
Ctrl + e : move to the end of line.
Ctrl + d : if you've type something, Ctrl + d deletes the character under the cu
Ctrl + k : delete all text from the cursor to the end of line.
Ctrl + x + backspace : delete all text from the beginning of line to the cursor
Ctrl + t : transpose the character before the cursor with the one under the curs
Ctrl + w : cut the word before the cursor; then Ctrl + y paste it
Ctrl + u : cut the line before the cursor; then Ctrl + y paste it
Ctrl + _ : undo typing.
Ctrl + l : equivalent to clear.
Ctrl + x + Ctrl + e : launch editor defined by $EDITOR to input your command. Us
```

Change case

```
Esc + u
# converts text from cursor to the end of the word to uppercase.
Esc + l
# converts text from cursor to the end of the word to lowercase.
Esc + c
# converts letter under the cursor to uppercase.
```

Run history number (e.g. 53)

```
!53
```

Run last command

```
!!
# run the previous command using sudo
sudo !!
# of course you need to enter your password
```

Run last command and change some parameter using caret substitution (e.g. last command: echo 'aaa' -> rerun as: echo 'bbb')

```
#last command: echo 'aaa'
^aaa^bbb

#echo 'bbb'
#bbb

#Notice that only the first aaa will be replaced, if you want to replace all 'aa
^aaa^bbb^:&
#or
!!:gs/aaa/bbb/
```

Run past command that began with (e.g. cat filename)

```
!cat
# or
!c
# run cat filename again
```

Bash globbing

```
# '*' serves as a "wild card" for filename expansion.
/etc/pa*wd    #/etc/passwd

# '?' serves as a single-character "wild card" for filename expansion.
/b?n/?at      #/bin/cat

# '[]' serves to match the character from a range.
ls -l [a-z]*   #list all files with alphabet in its filename.

# '{}' can be used to match filenames with more than one patterns
ls {*.sh,*.py}   #list all .sh and .py files
```

Some handy environment variables

```
$0   :name of shell or shell script.
$1, $2, $3, ... :positional parameters.
$#   :number of positional parameters.
$?   :most recent foreground pipeline exit status.
$-   :current options set for the shell.
$$   :pid of the current shell (not subshell).
$!   :is the PID of the most recent background command.

$DESKTOP_SESSION     current display manager
$EDITOR   preferred text editor.
$LANG   current language.
$PATH   list of directories to search for executable files (i.e. ready-to-run pr
$PWD    current directory
$SHELL  current shell
$USER   current username
$HOSTNAME   current hostname
```

Variable

[back to top]

Variable substitution within quotes

```
# foo=bar
 echo "'$foo'"
#'bar'
# double/single quotes around single quotes make the inner single quotes expand
```

Get the length of variable

```
var="some string"
echo ${#var}
# 11
```

Get the first character of the variable

```
var=string
echo "${var:0:1}"
#s

# or
echo ${var%%"${var#?}"}
```

Remove the first or last string from variable

```
var="some string"
echo ${var:2}
#me string
```

Replacement (e.g. remove the first leading 0)

```
var="0050"
echo ${var[@]#0}
#050
```

Replacement (e.g. replace 'a' with ',')

```
{var/a/,}
```

Replace all (e.g. replace all 'a' with ',')

```
{var//a/,}
```

```
#with grep
 test="god the father"
 grep ${test// /\\\|} file.txt
 # turning the space into 'or' (\|) in grep
```

To change the case of the string stored in the variable to lowercase (Parameter Expansion)

```
var=HelloWorld
echo ${var,,}
helloworld
```

Expand and then execute variable/argument

```
cmd="bar=foo"
eval "$cmd"
echo "$bar" # foo
```

Math

[back to top]

Arithmetic Expansion in Bash (Operators: +, -, *, /, %, etc)

```
echo $(( 10 + 5 ))  #15
x=1
echo $(( x++ )) #1 , notice that it is still 1, since it's post-incremen
echo $(( x++ )) #2
echo $(( ++x )) #4 , notice that it is not 3 since it's pre-incremen
echo $(( x-- )) #4
echo $(( x-- )) #3
echo $(( --x )) #1
x=2
y=3
echo $(( x ** y )) #8
```

Print out the prime factors of a number (e.g. 50)

```
factor 50
# 50: 2 5 5
```

Sum up input list (e.g. seq 10)

```
seq 10|paste -sd+|bc
```

Sum up a file (each line in file contains only one number)

```
awk '{s+=$1} END {print s}' filename
```

Column subtraction

```
cat file| awk -F '\t' 'BEGIN {SUM=0}{SUM+=$3-$2}END{print SUM}'
```

Simple math with expr

```
expr 10+20 #30
expr 10\*20 #600
expr 30 \> 20 #1 (true)
```

More math with bc

```
# Number of decimal digit/ significant figure
echo "scale=2;2/3" | bc
#.66

# Exponent operator
echo "10^2" | bc
#100

# Using variables
echo "var=5;--var"| bc
#4
```

Grep

[back to top]

Type of grep

```
grep = grep -G # Basic Regular Expression (BRE)
fgrep = grep -F # fixed text, ignoring meta-charachetrs
egrep = grep -E # Extended Regular Expression (ERE)
pgrep = grep -P # Perl Compatible Regular Expressions (PCRE)
rgrep = grep -r # recursive
```

Grep and count number of empty lines

```
grep -c "^$"
```

Grep and return only integer

```
grep -o '[0-9]*'
#or
grep -oP '\d'
```

Grep integer with certain number of digits (e.g. 3)

```
grep ‘[0-9]\{3\}’
# or
grep -E ‘[0-9]{3}’
```

```
# or
grep -P '\d{3}'
```

Grep only IP address

```
grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
# or
grep -Po '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
```

Grep whole word (e.g. 'target')

```
grep -w 'target'

#or using RE
grep '\btarget\b'
```

Grep returning lines before and after match (e.g. 'bbo')

```
# return also 3 lines after match
grep -A 3 'bbo'

# return also 3 lines before match
grep -B 3 'bbo'

# return also 3 lines before and after match
grep -C 3 'bbo'
```

Grep string starting with (e.g. 'S')

```
grep -o 'S.*'
```

Extract text between words (e.g. w1,w2)

```
grep -o -P '(?<=w1).*(?=w2)'
```

Grep lines without word (e.g. 'bbo')

```
grep -v bbo filename
```

Grep lines not begin with string (e.g. #)

```
grep -v '^#' file.txt
```

Grep variables with space within it (e.g. myvar="some strings")

```
grep "$myvar" filename
#remember to quote the variable!
```

Grep only one/first match (e.g. 'bbo')

```
grep -m 1 bbo filename
```

Grep and return number of matching line(e.g. 'bbo')

```
grep -c bbo filename
```

Count occurrence (e.g. three times a line count three times)

```
grep -o bbo filename |wc -l
```

Case insensitive grep (e.g. 'bbo'/'BBO'/'Bbo')

```
grep -i "bbo" filename
```

COLOR the match (e.g. 'bbo')!

```
grep --color bbo filename
```

Grep search all files in a directory(e.g. 'bbo')

```
grep -R bbo /path/to/directory
# or
grep -r bbo /path/to/directory
```

Search all files in directory, do not ouput the filenames (e.g. 'bbo')

```
grep -rh bbo /path/to/directory
```

Search all files in directory, output ONLY the filenames with matches(e.g. 'bbo')

```
grep -rl bbo /path/to/directory
```

Grep OR (e.g. A or B or C or D)

```
grep 'A\|B\|C\|D'
```

Grep AND (e.g. A and B)

```
grep 'A.*B'
```

Regex any singer character (e.g. ACB or AEB)

```
grep 'A.B'
```

Regex with or without a certain character (e.g. color or colour)

```
grep 'colou?r'
```

Grep all content of a fileA from fileB

```
grep -f fileA fileB
```

Grep a tab

```
grep $'\t'
```

Grep variable from variable

```
$echo "$long_str"|grep -q "$short_str"
if [ $? -eq 0 ]; then echo 'found'; fi
#grep -q will output 0 if match found
#remember to add space between []!
```

Grep strings between a bracket()

```
grep -oP '\(\K[^\)]+'
```

Grep number of characters with known strings in between(e.g. AAEL000001-RA)

```
grep -o -w "\w\{10\}\-R\w\{1\}"
# \w word character [0-9a-zA-Z_] \W not word character
```

Skip directory (e.g. 'bbo')

```
grep -d skip 'bbo' /path/to/files/*
```

Sed

[back to top]

Remove the 1st line

```
sed 1d filename
```

Remove the first 100 lines (remove line 1-100)

```
sed 1,100d filename
```

Remove lines with string (e.g. 'bbo')

```
sed "/bbo/d" filename
# case insensitive:
sed "/bbo/Id" filename
```

Remove lines whose nth character not equal to a value (e.g. 5th character not equal to 2)

```
sed -E '/^.{5}[^2]/d'
#aaaa2aaa (you can stay)
#aaaa1aaa (delete!)
```

Edit infile (edit and save to file), (e.g. deleting the lines with 'bbo' and save to file)

```
sed -i "/bbo/d" filename
```

When using variable (e.g. $i), use double quotes " "

```
# e.g. add >$i to the first line (to make a bioinformatics FASTA file)
sed "1i >$i"
# notice the double quotes! in other examples, you can use a single quote, but
# '1i' means insert to first line
```

Using environment variable and end-of-line pattern at the same time.

```
# Use backslash for end-of-line $ pattern, and double quotes for expressing the
sed -e "\$s/\$/\n+--$3-----+/"
```

Delete/remove empty lines

```
sed '/^\s*$/d'

# or

sed '/^$/d'
```

Delete/remove last line

```
sed '$d'
```

Delete/remove last character from end of file

```
sed -i '$ s/.$//' filename
```

Add string to beginning of file (e.g. "[")

```
sed -i '1s/^/[/' file
```

Add string at certain line number (e.g. add 'something' to line 1 and line 3)

```
sed -e '1isomething -e '3isomething'
```

Add string to end of file (e.g. "]")

```
sed '$s/$/]/' filename
```

Add newline to the end

```
sed '$a\'
```

Add string to beginning of every line (e.g. 'bbo')

```
sed -e 's/^/bbo/' file
```

Add string to end of each line (e.g. "}")

```
sed -e 's/$/\}\]/' filename
```

Add \n every nth character (e.g. every 4th character)

```
sed 's/.\{4\}/&\n/g'
```

Concatenate/combine/join files with a seperator and next line (e.g separate by ",")

```
sed -s '$a,' *.json > all.json
```

Substitution (e.g. replace A by B)

```
sed 's/A/B/g' filename
```

Substitution with wildcard (e.g. replace a line start with aaa= by aaa=/my/new/path)

```
sed "s/aaa=.*/aaa=\/my\/new\/path/g"
```

Select lines start with string (e.g. 'bbo')

```
sed -n '/^@S/p'
```

Delete lines with string (e.g. 'bbo')

```
sed '/bbo/d' filename
```

Print/get/trim a range of line (e.g. line 500-5000)

```
sed -n 500,5000p filename
```

Print every nth lines

```
sed -n '0~3p' filename

# catch 0: start; 3: step
```

Print every odd # lines

```
sed -n '1~2p'
```

Print every third line including the first line

```
sed -n '1p;0~3p'
```

Remove leading whitespace and tabs

```
sed -e 's/^[ \t]*//'
# Notice a whitespace before '\t'!!
```

Remove only leading whitespace

```
sed 's/ *//'

# notice a whitespace before '*'!!
```

Remove ending commas

```
sed 's/,$//g'
```

Add a column to the end

```
sed "s/$/\t$i/"
# $i is the valuable you want to add

# To add the filename to every last column of the file
for i in $(ls);do sed -i "s/$/\t$i/" $i;done
```

Add extension of filename to last column

```
for i in T000086_1.02.n T000086_1.02.p;do sed "s/$/\t${i/*./}/" $i;done >T00008
```

Remove newline\ nextline

```
sed ':a;N;$!ba;s/\n//g'
```

Print a particular line (e.g. 123th line)

```
sed -n -e '123p'
```

Print a number of lines (e.g. line 10th to line 33 rd)

```
sed -n '10,33p' <filename
```

Change delimiter

```
sed 's=/=\\/=g'
```

Replace with wildcard (e.g A-1-e or A-2-e or A-3-e....)

```
sed 's/A-.*-e//g' filename
```

Remove last character of file

```
sed '$ s/.$//'
```

Insert character at specified position of file (e.g. AAAAAA –> AAA#AAA)

```
sed -r -e 's/^.{3}/&#/' file
```

Awk

[back to top]

Set tab as field separator

```
awk -F $'\t'
```

Output as tab separated (also as field separator)

```
awk -v OFS='\t'
```

Pass variable

```
a=bbo;b=obb;
awk -v a="$a" -v b="$b" "$1==a && $10=b" filename
```

Print line number and number of characters on each line

```
awk '{print NR,length($0);}' filename
```

Find number of columns

```
awk '{print NF}'
```

Reverse column order

```
awk '{print $2, $1}'
```

Check if there is a comma in a column (e.g. column $1)

```
awk '$1~/,/ {print}'
```

Split and do for loop

```
awk '{split($2, a,",");for (i in a) print $1"\t"a[i]}' filename
```

Print all lines before nth occurrence of a string (e.g stop print lines when 'bbo' appears 7 times)

```
awk -v N=7 '{print}/bbo/&& --N<=0 {exit}'
```

Print filename and last line of all files in directory

```
ls|xargs -n1 -I file awk '{s=$0};END{print FILENAME,s}' file
```

Add string to the beginning of a column (e.g add "chr" to column $3)

```
awk 'BEGIN{OFS="\t"}$3="chr"$3'
```

Remove lines with string (e.g. 'bbo')

```
awk '!/bbo/' file
```

Remove last column

```
awk 'NF{NF-=1};1' file
```

Usage and meaning of NR and FNR

```
# For example there are two files:
# fileA:
```

```
# a
# b
# c
# fileB:
# d
# e
awk 'print FILENAME, NR,FNR,$0}' fileA fileB
# fileA    1    1    a
# fileA    2    2    b
# fileA    3    3    c
# fileB    4    1    d
# fileB    5    2    e
```

AND gate

```
# For example there are two files:
# fileA:
# 1    0
# 2    1
# 3    1
# 4    0
# fileB:
# 1    0
# 2    1
# 3    0
# 4    1

awk -v OFS='\t' 'NR=FNR{a[$1]=$2;next} NF {print $1,((a[$1]=$2)? $2:"0")}' file
# 1    0
# 2    1
# 3    0
# 4    0
```

Round all numbers of file (e.g. 2 significant figure)

```
awk '{while (match($0, /[0-9]+\[0-9]+/)){
    \printf "%s%.2f", substr($0,0,RSTART-1),substr($0,RSTART,RLENGTH)
    \$0=substr($0, RSTART+RLENGTH)
    \}
    \print
    \}'
```

Give number/index to every row

```
awk '{printf("%s\t%s\n",NR,$0)}'
```

Break combine column data into rows

```
# For example, seperate the following content:
# David    cat,dog
# into
# David    cat
# David    dog

awk '{split($2,a,",");for(i in a)print $1"\t"a[i]}' file

# Detail here:  http://stackoverflow.com/questions/33408762/bash-turning-single-
```

Average a file (each line in file contains only one number)

```
awk '{s+=$1}END{print s/NR}'
```

Print field start with string (e.g Linux)

```
awk '$1 ~ /^Linux/'
```

Sort a row (e.g. 1 40 35 12 23 –> 1 12 23 35 40)

```
awk ' {split( $0, a, "\t" ); asort( a ); for( i = 1; i <= length(a); i++ ) print
```

Subtract previous row values (add column6 which equal to column4 minus last column5)

```
awk '{$6 = $4 - prev5; prev5 = $5; print;}'
```

Xargs

[back to top]

Set tab as delimiter (default:space)

```
xargs -d\t
```

Prompt commands before running commands

```
ls|xargs -L1 -p head
```

Display 3 items per line

```
echo 1 2 3 4 5 6| xargs -n 3
# 1 2 3
# 4 5 6
```

Prompt before execution

```
echo a b c |xargs -p -n 3
```

Print command along with output

```
xargs -t abcd
# bin/echo abcd
# abcd
```

With find and rm

```
find . -name "*.html"|xargs rm

# when using a backtick
rm `find . -name "*.html"`
```

Delete files with whitespace in filename (e.g. “hello 2001”)

```
find . -name "*.c" -print0|xargs -0 rm -rf
```

Show limits on command-line length

```
xargs --show-limits
# Output from my Ubuntu:
# Your environment variables take up 3653 bytes
# POSIX upper limit on argument length (this system): 2091451
```

```
# POSIX smallest allowable upper limit on argument length (all systems): 4096
# Maximum length of command we could actually use: 2087798
# Size of command buffer we are actually using: 131072
# Maximum parallelism (--max-procs must be no greater): 2147483647
```

Move files to folder

```
find . -name "*.bak" -print 0|xargs -0 -I {} mv {} ~/old

# or
find . -name "*.bak" -print 0|xargs -0 -I file mv file ~/old
```

Move first 100th files to a directory (e.g. d1)

```
ls |head -100|xargs -I {} mv {} d1
```

Parallel

```
time echo {1..5} |xargs -n 1 -P 5 sleep

# a lot faster than:
time echo {1..5} |xargs -n1 sleep
```

Copy all files from A to B

```
find /dir/to/A -type f -name "*.py" -print 0| xargs -0 -r -I file cp -v -p file

# v: verbose|
# p: keep detail (e.g. owner)
```

With sed

```
ls |xargs -n1 -I file sed -i '/^Pos/d' file
```

Add the file name to the first line of file

```
ls |sed 's/.txt//g'|xargs -n1 -I file sed -i -e '1 i\>file\' file.txt
```

Count all files

```
ls |xargs -n1 wc -l
```

Turn output into a single line

```
ls -l| xargs
```

Count files within directories

```
echo mso{1..8}|xargs -n1 bash -c 'echo -n "$1:"; ls -la "$1"| grep -w 74 |wc -l
# "--" signals the end of options and display further option processing
```

Count lines in all file, also count total lines

```
ls|xargs wc -l
```

Xargs and grep

```
cat grep_list |xargs -I{} grep {} filename
```

Xargs and sed (replace all old ip address with new ip address under /etc directory)

```
grep -rl '192.168.1.111' /etc | xargs sed -i 's/192.168.1.111/192.168.2.111/g'
```

Find

[back to top]

List all sub directory/file in the current directory

```
find .
```

List all files under the current directory

```
find . -type f
```

List all directories under the current directory

```
find . -type d
```

Edit all files under current directory (e.g. replace 'www' with 'ww')

```
find . -name '*.php' -exec sed -i 's/www/w/g' {} \;

# if there are no subdirectory
replace "www" "w" -- *
# a space before *
```

Find and output only filename (e.g. "mso")

```
find mso*/ -name M* -printf "%f\n"
```

Find large files in the system (e.g. >4G)

```
find / -type f -size +4G
```

Find and delete file with size less than (e.g. 74 byte)

```
find . -name "*.mso" -size -74c -delete

# M for MB, etc
```

Find empty (0 byte) files

```
find . -type f -empty
# to further delete all the empty files
find . -type f -empty -delete
```

Recursively count all the files in a directory

```
find . -type f | wc -l
```

Condition and loop

[back to top]

If statement

```
# if and else loop for string matching
if [[ "$c" == "read" ]]; then outputdir="seq"; else outputdir="write" ; fi

# Test if myfile contains the string 'test':
if grep -q hello myfile; then echo -e "file contains the string!" ; fi

# Test if mydir is a directory, change to it and do other stuff:
if cd mydir; then
  echo 'some content' >myfile
else
  echo >&2 "Fatal error. This script requires mydir."
fi

# if variable is null
if [ ! -s "myvariable" ]; then echo -e "variable is null!" ; fi
#True of the length if "STRING" is zero.

# Using test command (same as []), to test if the length of variable is nonzero
test -n "$myvariable" && echo myvariable is "$myvariable" || echo myvariable is

# Test if file exist
if [ -e 'filename' ]
then
  echo -e "file exists!"
fi

# Test if file exist but also including symbolic links:
if [ -e myfile ] || [ -L myfile ]
then
  echo -e "file exists!"
fi

# Test if the value of x is greater or equal than 5
if [ "$x" -ge 5 ]; then echo -e "greater or equal than 5!" ; fi

# Test if the value of x is greater or equal than 5, in bash/ksh/zsh:
if ((x >= 5)); then echo -e "greater or equal than 5!" ; fi

# Use (( )) for arithmetic operation
if ((j==u+2)); then echo -e "j==u+2!!" ; fi

# Use [[ ]] for comparison
if [[ $age -gt 21 ]]; then echo -e "forever 21!!" ; fi
```

More if commands

For loop

```
# Echo the file name under the current directory
for i in $(ls); do echo file $i;done
#or
for i in *; do echo file $i; done

# Make directories listed in a file (e.g. myfile)
for dir in $(<myfile); do mkdir $dir; done

# Press any key to continue each loop
for i in $(cat tpc_stats_0925.log |grep failed|grep -o '\query\w\{1,2\}');do cat

# Print a file line by line when a key is pressed,
oifs="$IFS"; IFS=$'\n'; for line in $(cat myfile); do ...; done
while read -r line; do ...; done <myfile

#If only one word a line, simply
for line in $(cat myfile); do echo $line; read -n1; done

#Loop through an array
for i in "${arrayName[@]}"; do echo $i;done
```

While loop,

```
# Column subtraction of a file (e.g. a 3 columns file)
while read a b c; do echo $(($c-$b));done < <(head filename)
#there is a space between the two '<'s

# Sum up column subtraction
i=0; while read a b c; do ((i+=$c-$b)); echo $i; done < <(head filename)

# Keep checking a running process (e.g. perl) and start another new process (e.g
while [[ $(pidof perl) ]];do echo f;sleep 10;done && python timetorunpython.py
```

switch (case in bash)

```
read type;
case $type in
  '0')
    echo 'how'
    ;;
  '1')
    echo 'are'
```

```
    ;;
  '2')
    echo 'you'
    ;;
esac
```

Time

[back to top]

Find out the time require for executing a command

```
time echo hi
```

Wait for some time (e.g 10s)

```
sleep 10
```

Print date with formatting

```
date +%F
# 2020-07-19

# or
date +'%d-%b-%Y-%H:%M:%S'
# 10-Apr-2020-21:54:40

# Returns the current time with nanoseconds.
date +"%T.%N"
# 11:42:18.664217000

# Get the seconds since epoch (Jan 1 1970) for a given date (e.g Mar 16 2021)
date -d "Mar 16 2021" +%s
# 1615852800
# or
date -d "Tue Mar 16 00:00:00 UTC 2021"  +%s
# 1615852800

# Convert the number of seconds since epoch back to date
date --date @1615852800
# Tue Mar 16 00:00:00 UTC 2021
```

wait for random duration (e.g. sleep 1-5 second, like adding a jitter)

```
sleep $[ ( $RANDOM % 5 ) + 1 ]
```

Log out your account after a certain period of time (e.g 10 seconds)

```
TMOUT=10
#once you set this variable, logout timer start running!
```

Set how long you want to run a command

```
#This will run the command 'sleep 10' for only 1 second.
timeout 1 sleep 10
```

Set when you want to run a command (e.g 1 min from now)

```
at now + 1min  #time-units can be minutes, hours, days, or weeks
warning: commands will be executed using /bin/sh
at> echo hihigithub >~/itworks
at> <EOT>   # press Ctrl + D to exit
job 1 at Wed Apr 18 11:16:00 2018
```

Download

[back to top]

Download the content of this README.md (the one your are viewing now)

```
curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/README.md

# or w3m (a text based web browser and pager)
curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/README.md

# or using emacs (in emac text editor)
emacs --eval '(org-mode)' --insert <(curl https://raw.githubusercontent.com/onc

# or using emacs (on terminal, exit using Ctrl + x then Ctrl + c)
emacs -nw --eval '(org-mode)' --insert <(curl https://raw.githubusercontent.com/
```

Download all from a page

```
wget -r -l1 -H -t1 -nd -N -np -A mp3 -e robots=off http://example.com

# -r: recursive and download all links on page
# -l1: only one level link
# -H: span host, visit other hosts
# -t1: numbers of retries
# -nd: don't make new directories, download to here
# -N: turn on timestamp
# -nd: no parent
# -A: type (separate by ,)
# -e robots=off: ignore the robots.txt file which stop wget from crashing the s
```

Upload a file to web and download (https://transfer.sh/)

```
#  Upload a file (e.g. filename.txt):
curl --upload-file ./filename.txt https://transfer.sh/filename.txt
# the above command will return a URL, e.g: https://transfer.sh/tG8rM/filename.

# Next you can download it by:
curl https://transfer.sh/tG8rM/filename.txt -o filename.txt
```

Download file if necessary

```
data=file.txt
url=http://www.example.com/$data
if [ ! -s $data ];then
    echo "downloading test data..."
    wget $url
fi
```

Wget to a filename (when a long name)

```
wget -O filename "http://example.com"
```

Wget files to a folder

```
wget -P /path/to/directory "http://example.com"
```

Instruct curl to follow any redirect until it reaches the final destination:

```
curl -L google.com
```

Random

[back to top]

Random generate password (e.g. generate 5 password each of length 13)

```
sudo apt install pwgen
pwgen 13 5
#sahcahS9dah4a xieXaiJaey7xa UuMeo0ma7eic9 Ahpah9see3zai acerae7Huigh7
```

Random pick 100 lines from a file

```
shuf -n 100 filename
```

Random order (lucky draw)

```
for i in a b c d e; do echo $i; done | shuf
```

Echo series of random numbers between a range (e.g. shuffle numbers from 0-100, then pick 15 of them randomly)

```
shuf -i 0-100 -n 15
```

Echo a random number

```
echo $RANDOM
```

Random from 0-9

```
echo $((RANDOM % 10))
```

Random from 1-10

```
echo $(((RANDOM %10)+1))
```

Xwindow

[back to top]

X11 GUI applications! Here are some GUI tools for you if you get bored by the text-only environment.

Enable X11 forwarding,in order to use graphical application on servers

```
ssh -X user_name@ip_address

# or setting through xhost
# --> Install the following for Centos:
# xorg-x11-xauth
# xorg-x11-fonts-*
# xorg-x11-utils
```

Little xwindow tools

```
xclock
xeyes
xcowsay
```

Open pictures/images from ssh server

```
1. ssh -X user_name@ip_address
2. apt-get install eog
3. eog picture.png
```

Watch videos on server

```
1. ssh -X user_name@ip_address
2. sudo apt install mpv
3. mpv myvideo.mp4
```

Use gedit on server (GUI editor)

```
1. ssh -X user_name@ip_address
2. apt-get install gedit
3. gedit filename.txt
```

Open PDF file from ssh server

```
1. ssh -X user_name@ip_address
2. apt-get install evince
3. evince filename.pdf
```

Use google-chrome browser from ssh server

```
1. ssh -X user_name@ip_address
2. apt-get install libxss1 libappindicator1 libindicator7
3. wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.de
4. sudo apt-get install -f
5. dpkg -i google-chrome*.deb
6. google-chrome
```

System

[back to top]

Work with yum history

```
# List yum history (e.g install, update)
sudo yum history
# Example output:
# Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
# ID     | Login user               | Date and time    | Action(s)      | Altere
# ---------------------------------------------------------------------------
#     11 |  ... <myuser>       | 2020-04-10 10:57 | Install        |    1 P<
#     10 |  ... <myuser>       | 2020-03-27 05:21 | Install        |    1 >P
#      9 |  ... <myuser>       | 2020-03-05 11:57 | I, U           |   56 *<
# ...

# Show more details of a yum history (e.g. history #11)
sudo yum history info 11

# Undo a yum history (e.g. history #11, this will uninstall some packages)
sudo yum history undo 11
```

Audit files to see who made changes to a file [RedHat based system only]

```
# To audit a directory recursively for changes (e.g. myproject)
auditctl -w /path/to/myproject/ -p wa

# If you delete a file name "VIPfile", the deletion is recorded in /var/log/audi
sudo grep VIPfile /var/log/audit/audit.log
#type=PATH msg=audit(1581417313.678:113): item=1 name="VIPfile" inode=300115 dev
```

Check out whether SELinux is enabled

```
sestatus
# SELinux status:                 enabled
# SELinuxfs mount:                /sys/fs/selinux
# SELinux root directory:         /etc/selinux
# Loaded policy name:             targeted
# Current mode:                   enforcing
# Mode from config file:          enforcing
# Policy MLS status:              enabled
# Policy deny_unknown status:     allowed
# Max kernel policy version:      31
```

Generate public key from private key

```
ssh-keygen -y -f ~/.ssh/id_rsa > ~/.ssh/id_rsa.pub
```

Copy your default public key to remote user

```
ssh-copy-id <user_name>@<server_IP>
# then you need to enter the password
# and next time you won't need to enter password when ssh to that user
```

Copy default public key to remote user using the required private key (e.g. use your mykey.pem key to copy your id_rsa.pub to the remote user)

```
# before you need to use mykey.pem to ssh to remote user.
ssh-copy-id -i ~/.ssh/id_rsa.pub -o "IdentityFile ~/Downloads/mykey.pem" <user_
# now you don't need to use key to ssh to that user.
```

SSH Agent Forwarding

```
# To bring your key with you when ssh to serverA, then ssh to serverB from serve
ssh-agent
ssh-add /path/to/mykey.pem
ssh -A <username>@<IP_of_serverA>
# Next you can ssh to serverB
ssh <username>@<IP_of_serverB>
```

Set the default user and key for a host when using SSH

```
# add the following to ~/.ssh/config
Host myserver
  User myuser
  IdentityFile ~/path/to/mykey.pem

# Next, you could run "ssh myserver" instead of "ssh -i ~/path/to/mykey.pem myus
```

Follow the most recent logs from service

```
journalctl -u <service_name> -f
```

Eliminate the zombie

```
# A zombie is already dead, so you cannot kill it. You can eliminate the zombie
# First, find PID of the zombie
ps aux| grep 'Z'
# Next find the PID of zombie's parent
pstree -p -s <zombie_PID>
# Then you can kill its parent and you will notice the zombie is gone.
sudo kill 9 <parent_PID>
```

Show memory usage

```
free -c 10 -mhs 1
# print 10 times, at 1 second interval
```

Display CPU and IO statistics for devices and partitions.

```
# refresh every second
iostat -x -t 1
```

Display bandwidth usage on an network interface (e.g. enp175s0f0)

```
iftop -i enp175s0f0
```

Tell how long the system has been running and number of users

```
uptime
```

Check if it's root running

```
if [ "$EUID" -ne 0 ]; then
        echo "Please run this as root"
        exit 1
fi
```

Change shell of a user (e.g. bonnie)

```
chsh -s /bin/sh bonnie
# /etc/shells: valid login shells
```

Change root / fake root / jail (e.g. change root to newroot)

```
chroot /home/newroot /bin/bash

# To exit chroot
exit
```

Display file status (size; access, modify and change time, etc) of a file (e.g. filename.txt)

```
stat filename.txt
```

Snapshot of the current processes

```
ps aux
```

Display a tree of processes

```
pstree
```

Find maximum number of processes

```
cat /proc/sys/kernel/pid_max
```

Print or control the kernel ring buffer

```
dmesg
```

Show IP address

```
$ip add show

# or
ifconfig
```

Print previous and current SysV runlevel

```
runlevel

# or
who -r
```

Change SysV runlevel (e.g. 5)

```
init 5
#or
telinit 5
```

Display all available services in all runlevels,

```
chkconfig --list
# update-rc.d equivalent to chkconfig in ubuntu
```

Check system version

```
cat /etc/*-release
```

Linux Programmer's Manuel: hier- description of the filesystem hierarchy

```
man hier
```

Control the systemd system and service manager

```
# e.g. check the status of cron service
systemctl status cron.service

# e.g. stop cron service
systemctl stop cron.service
```

List job

```
jobs -l
```

Run a program with modified priority (e.g. ./test.sh)

```
# nice value is adjustable from -20 (most favorable) to +19
# the nicer the application, the lower the priority
# Default niceness: 10; default priority: 80

nice -10 ./test.sh
```

Export PATH

```
export PATH=$PATH:~/path/you/want
```

Make file executable

```
chmod +x filename
# you can now ./filename to execute it
```

Print system information

```
uname -a

# Check system hardware-platform (x86-64)
uname -i
```

Surf the net

```
links www.google.com
```

Add user, set passwd

```
useradd username
passwd username
```

Edit PS1 variable for bash (e.g. displaying the whole path)

```
1. vi ~/.bash_profile
2. export PS1='\u@\h:\w\$'
# $PS1 is a variable that defines the makeup and style of the command prompt
# You could use emojis and add timestamp to every prompt using the following va
# export PS1="\t@🦁:\w\$ "
3. source ~/.bash_profile
```

Edit environment setting (e.g. alias)

```
1. vi ~/.bash_profile
2. alias pd="pwd" //no more need to type that 'w'!
3. source ~/.bash_profile
```

Print all alias

```
alias -p
```

Unalias (e.g. after alias ls='ls –color=auto')

```
unalias ls
```

Set and unset shell options

```
# print all shell options
shopt

# to unset (or stop) alias
shopt -u expand_aliases

# to set (or start) alias
shopt -s expand_aliases
```

List environment variables (e.g. PATH)

```
echo $PATH
# list of directories separated by a colon
```

List all environment variables for current user

```
env
```

Unset environment variable (e.g. unset variable 'MYVAR')

```
unset MYVAR
```

Show partition format

```
lsblk
```

Inform the OS of partition table changes

```
partprobe
```

Soft link program to bin

```
ln -s /path/to/program /home/usr/bin
# must be the whole path to the program
```

Show hexadecimal view of data

```
hexdump -C filename.class
```

Jump to different node

```
rsh node_name
```

Check port (active internet connection)

```
netstat -tulpn
```

Print resolved symbolic links or canonical file names

```
readlink filename
```

Find out the type of command and where it link to (e.g. python)

```
type python
# python is /usr/bin/python
# There are 5 different types, check using the 'type -f' flag
# 1. alias    (shell alias)
# 2. function (shell function, type will also print the function body)
# 3. builtin  (shell builtin)
# 4. file     (disk file)
# 5. keyword  (shell reserved word)

# You can also use `which`
which python
# /usr/bin/python
```

List all functions names

```
declare -F
```

List total size of a directory

```
du -hs .

# or
du -sb
```

Copy directory with permission setting

```
cp -rp /path/to/directory
```

Store current directory

```
pushd .

# then pop
popd

#or use dirs to display the list of currently remembered directories.
dirs -l
```

Show disk usage

```
df -h

# or
du -h

#or
du -sk /var/log/* |sort -rn |head -10
```

check the Inode utilization

```
df -i
# Filesystem      Inodes  IUsed   IFree IUse% Mounted on
# devtmpfs        492652    304  492348    1% /dev
# tmpfs           497233      2  497231    1% /dev/shm
# tmpfs           497233    439  496794    1% /run
# tmpfs           497233     16  497217    1% /sys/fs/cgroup
# /dev/nvme0n1p1 5037976 370882 4667094    8% /
# tmpfs           497233      1  497232    1% /run/user/1000
```

Show all file system type

```
df -TH
```

Show current runlevel

```
runlevel
```

Switch runlevel

```
init 3

#or
telinit 3
```

Permanently modify runlevel

```
1. edit /etc/init/rc-sysinit.conf
2. env DEFAULT_RUNLEVEL=2
```

Become root

```
su
```

Become somebody

```
su somebody
```

Report user quotes on device

```
repquota -auvs
```

Get entries in a number of important databases

```
getent database_name

# (e.g. the 'passwd' database)
getent passwd
# list all user account (all local and LDAP)
```

```
# (e.g. fetch list of grop accounts)
getent group
# store in database 'group'
```

Change owner of file

```
chown user_name filename
chown -R user_name /path/to/directory/
# chown user:group filename
```

Mount and unmount

```
# e.g. Mount /dev/sdb to /home/test
mount /dev/sdb /home/test

# e.g. Unmount /home/test
umount /home/test
```

List current mount detail

```
mount
# or
df
```

List current usernames and user-numbers

```
cat /etc/passwd
```

Get all username

```
getent passwd| awk '{FS="[:]"; print $1}'
```

Show all users

```
compgen -u
```

Show all groups

```
compgen -g
```

Show group of user

```
group username
```

Show uid, gid, group of user

```
id username

# variable for UID
echo $UID
```

Check if it's root

```
if [ $(id -u) -ne 0 ];then
    echo "You are not root!"
    exit;
fi
# 'id -u' output 0 if it's not root
```

Find out CPU information

```
more /proc/cpuinfo

# or
lscpu
```

Set quota for user (e.g. disk soft limit: 120586240; hard limit: 125829120)

```
setquota username 120586240 125829120 0 0 /home
```

Show quota for user

```
quota -v username
```

Display current libraries from the cache

```
ldconfig -p
```

Print shared library dependencies (e.g. for 'ls')

```
ldd /bin/ls
```

Check user login

```
lastlog
```

Check last reboot history

```
last reboot
```

Edit path for all users

```
joe /etc/environment
# edit this file
```

Show and set user limit

```
ulimit -u
```

Print out number of cores/ processors

```
nproc --all
```

Check status of each core

```
1. top
2. press '1'
```

Show jobs and PID

```
jobs -l
```

List all running services

```
service --status-all
```

Schedule shutdown server

```
shutdown -r +5 "Server will restart in 5 minutes. Please save your work."
```

Cancel scheduled shutdown

```
shutdown -c
```

Broadcast to all users

```
wall -n hihi
```

Kill all process of a user

```
pkill -U user_name
```

Kill all process of a program

```
kill -9 $(ps aux | grep 'program_name' | awk '{print $2}')
```

Set gedit preference on server

```
# You might have to install the following:

apt-get install libglib2.0-bin;
# or
yum install dconf dconf-editor;
yum install dbus dbus-x11;

# Check list
gsettings list-recursively

# Change some settings
gsettings set org.gnome.gedit.preferences.editor highlight-current-line true
gsettings set org.gnome.gedit.preferences.editor scheme 'cobalt'
```

```
gsettings set org.gnome.gedit.preferences.editor use-default-font false
gsettings set org.gnome.gedit.preferences.editor editor-font 'Cantarell Regular
```

Add user to a group (e.g add user 'nice' to the group 'docker', so that he can run docker without sudo)

```
sudo gpasswd -a nice docker
```

Pip install python package without root

```
1. pip install --user package_name
2. You might need to export ~/.local/bin/ to PATH: export PATH=$PATH:~/.local/b
```

Removing old linux kernels (when /boot almost full...)

```
1. uname -a  #check current kernel, which should NOT be removed
2. sudo apt-get purge linux-image-X.X.X-X-generic  #replace old version
```

Change hostname

```
sudo hostname your-new-name

# if not working, do also:
hostnamectl set-hostname your-new-hostname
# then check with:
hostnamectl
# Or check /etc/hostname

# If still not working..., edit:
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-ensxxx
#add HOSTNAME="your-new-hostname"
```

List installed packages

```
apt list --installed

# or on Red Hat:
yum list installed
```

Check for package update

```
apt list --upgradeable

# or
sudo yum check-update
```

Run yum update excluding a package (e.g. do not update php packages)

```
sudo yum update --exclude=php*
```

Check which file make the device busy on umount

```
lsof /mnt/dir
```

When sound not working

```
killall pulseaudio
# then press Alt-F2 and type in pulseaudio
```

When sound not working

```
killall pulseaudio
```

List information about SCSI devices

```
lsscsi
```

Tutorial for setting up your own DNS server

http://onceuponmine.blogspot.tw/2017/08/set-up-your-own-dns-server.html

Tutorial for creating a simple daemon

http://onceuponmine.blogspot.tw/2017/07/create-your-first-simple-daemon.html

Tutorial for using your gmail to send email

http://onceuponmine.blogspot.tw/2017/10/setting-up-msmtprc-and-use-your-gmail.html

Using telnet to test open ports, test if you can connect to a port (e.g 53) of a server (e.g

192.168.2.106)

```
telnet 192.168.2.106 53
```

Change network maximum transmission unit (mtu) (e.g. change to 9000)

```
ifconfig eth0 mtu 9000
```

Get pid of a running process (e.g python)

```
pidof python

# or
ps aux|grep python
```

Check status of a process using PID

```
ps -p <PID>

#or
cat /proc/<PID>/status
cat /proc/<PID>/stack
cat /proc/<PID>/stat
```

NTP

```
# Start ntp:
ntpd

# Check ntp:
ntpq -p
```

Remove unnecessary files to clean your server

```
sudo apt-get autoremove
sudo apt-get clean
sudo rm -rf ~/.cache/thumbnails/*

# Remove old kernal:
sudo dpkg --list 'linux-image*'
```

```
sudo apt-get remove linux-image-OLDER_VERSION
```

Increase/ resize root partition (root partition is an LVM logical volume)

```
pvscan
lvextend -L +130G /dev/rhel/root -r
# Adding -r will grow filesystem after resizing the volume.
```

Create a UEFI Bootable USB drive (e.g. /dev/sdc1)

```
sudo dd if=~/path/to/isofile.iso of=/dev/sdc1 oflag=direct bs=1048576
```

Locate and remove a package

```
sudo dpkg -l | grep <package_name>
sudo dpkg --purge <package_name>
```

Create a ssh tunnel

```
ssh -f -L 9000:targetservername:8088 root@192.168.14.72 -N
#-f: run in background; -L: Listen; -N: do nothing
#the 9000 of your computer is now connected to the 8088 port of the targetservel
#so that you can see the content of targetservername:8088 by entering localhost
```

Get process ID of a process (e.g. sublime_text)

```
#pidof
pidof sublime_text

#pgrep, you don't have to type the whole program name
pgrep sublim

#pgrep, echo 1 if process found, echo 0 if no such process
pgrep -q sublime_text && echo 1 || echo 0

#top, takes longer time
top|grep sublime_text
```

Some benchmarking tools for your server

aio-stress - AIO benchmark.
bandwidth - memory bandwidth benchmark.
bonnie++ - hard drive and file system performance benchmark.
dbench - generate I/O workloads to either a filesystem or to a networked CIFS or NFS server.
dnsperf - authorative and recursing DNS servers.
filebench - model based file system workload generator.
fio - I/O benchmark.
fs_mark - synchronous/async file creation benchmark.
httperf - measure web server performance.
interbench - linux interactivity benchmark.
ioblazer - multi-platform storage stack micro-benchmark.
iozone - filesystem benchmark.
iperf3 - measure TCP/UDP/SCTP performance.
kcbench - kernel compile benchmark, compiles a kernel and measures the time it takes.
lmbench - Suite of simple, portable benchmarks.
netperf - measure network performance, test unidirectional throughput, and end-to-end latency.
netpipe - network protocol independent performance evaluator.
nfsometer - NFS performance framework.
nuttcp - measure network performance.
phoronix-test-suite - comprehensive automated testing and benchmarking platform.
seeker - portable disk seek benchmark.
siege - http load tester and benchmark.
sockperf - network benchmarking utility over socket API.
spew - measures I/O performance and/or generates I/O load.
stress - workload generator for POSIX systems.
sysbench - scriptable database and system performance benchmark.
tiobench - threaded IO benchmark.
unixbench - the original BYTE UNIX benchmark suite, provide a basic indicator of the performance of a Unix-like system.
wrk - HTTP benchmark.

Performance monitoring tool - sar

```
# installation
# It collects the data every 10 minutes and generate its report daily. crontab
yum install sysstat
systemctl start sysstat
systemctl enable sysstat

# show CPU utilization 5 times every 2 seconds.
sar 2 5
```

```
# show memory  utilization 5 times every 2 seconds.
sar -r 2 5

# show paging statistics 5 times every 2 seconds.
sar -B 2 5

# To generate all network statistic:
sar -n ALL

# reading SAR log file using -f
sar -f /var/log/sa/sa31|tail
```

Reading from journal file

```
journalctl --file ./log/journal/a90c18f62af546ccba02fa3734f00a04/system.journal
```

Show a listing of last logged in users.

```
lastb
```

Show a listing of current logged in users, print information of them

```
who
```

Show who is logged on and what they are doing

```
w
```

Print the user names of users currently logged in to the current host.

```
users
```

Stop tailing a file on program terminate

```
tail -f --pid=<PID> filename.txt
# replace <PID> with the process ID of the program.
```

List all enabled services

```
systemctl list-unit-files|grep enabled
```

Hardware

[back to top]

Collect and summarize all hardware info of your machine

```
lshw -json >report.json
# Other options are: [ -html ]  [ -short ]  [ -xml ]  [ -json ]  [ -businfo ]
```

Finding Out memory device detail

```
sudo dmidecode -t memory
```

Print detail of CPU hardware

```
dmidecode -t 4
#          Type   Information
#          0   BIOS
#          1   System
#          2   Base Board
#          3   Chassis
#          4   Processor
#          5   Memory Controller
#          6   Memory Module
#          7   Cache
#          8   Port Connector
#          9   System Slots
#         11   OEM Strings
#         13   BIOS Language
#         15   System Event Log
#         16   Physical Memory Array
#         17   Memory Device
#         18   32-bit Memory Error
#         19   Memory Array Mapped Address
#         20   Memory Device Mapped Address
#         21   Built-in Pointing Device
#         22   Portable Battery
#         23   System Reset
#         24   Hardware Security
#         25   System Power Controls
```

```
#         26   Voltage Probe
#         27   Cooling Device
#         28   Temperature Probe
#         29   Electrical Current Probe
#         30   Out-of-band Remote Access
#         31   Boot Integrity Services
#         32   System Boot
#         34   Management Device
#         35   Management Device Component
#         36   Management Device Threshold Data
#         37   Memory Channel
#         38   IPMI Device
#         39   Power Supply
```

Count the number of Segate hard disks

```
lsscsi|grep SEAGATE|wc -l
# or
sg_map -i -x|grep SEAGATE|wc -l
```

Get UUID of a disk (e.g. sdb)

```
lsblk -f /dev/sdb

# or
sudo blkid /dev/sdb
```

Generate an UUID

```
uuidgen
```

Print detail of all hard disks

```
lsblk -io KNAME,TYPE,MODEL,VENDOR,SIZE,ROTA
#where ROTA means rotational device / spinning hard disks (1 if true, 0 if false
```

List all PCI (Peripheral Component Interconnect) devices

```
lspci
# List information about NIC
lspci | egrep -i --color 'network|ethernet'
```

List all USB devices

```
lsusb
```

Linux modules

```
# Show the status of modules in the Linux Kernel
lsmod

# Add and remove modules from the Linux Kernel
modprobe

# or
# Remove a module
rmmod

# Insert a module
insmod
```

Controlling IPMI-enabled devices (e.g. BMC)

```
# Remotely finding out power status of the server
ipmitool -U <bmc_username> -P <bmc_password> -I lanplus -H <bmc_ip_address> powe

# Remotely switching on server
ipmitool -U <bmc_username> -P <bmc_password> -I lanplus -H <bmc_ip_address> powe

# Turn on panel identify light (default 15s)
ipmitool chassis identify 255

# Found out server sensor temperature
ipmitool sensors |grep -i Temp

# Reset BMC
ipmitool bmc reset cold

# Prnt BMC network
ipmitool lan print 1
```

```
# Setting BMC network
ipmitool -I bmc lan set 1 ipaddr 192.168.0.55
ipmitool -I bmc lan set 1 netmask 255.255.255.0
ipmitool -I bmc lan set 1 defgw ipaddr 192.168.0.1
```

Networking

[back to top]

Resolve a domain to IP address(es)

```
dig +short www.example.com

# or
host www.example.com
```

Get DNS TXT record a of domain

```
dig -t txt www.example.com

# or
host -t txt www.example.com
```

Send a ping with a limited TTL to 10 (TTL: Time-To-Live, which is the maximum number of hops that a packet can travel across the Internet before it gets discarded.)

```
ping 8.8.8.8 -t 10
```

Print the route packets trace to network host

```
traceroute google.com
```

Check connection to host (e.g. check connection to port 80 and 22 of google.com)

```
nc -vw5 google.com 80
# Connection to google.com 80 port [tcp/http] succeeded!

nc -vw5 google.com 22
# nc: connect to google.com port 22 (tcp) timed out: Operation now in progress
# nc: connect to google.com port 22 (tcp) failed: Network is unreachable
```

Nc as a chat tool!

```
# From server A:
$ sudo nc -l 80
# then you can connect to the 80 port from another server (e.g. server B):
# e.g. telent <server A IP address> 80
# then type something in server B
# and you will see the result in server A!
```

Check which ports are listening for TCP connections from the network

```
#notice that some companies might not like you using nmap
nmap -sT -O localhost

# check port 0-65535
nmap  -p0-65535 localhost
```

Check if a host is up and scan for open ports, also skip host discovery.

```
#skips checking if the host is alive which may sometimes cause a false positive
$ nmap google.com -Pn

# Example output:
# Starting Nmap 7.01 ( https://nmap.org ) at 2020-07-18 22:59 CST
# Nmap scan report for google.com (172.217.24.14)
# Host is up (0.013s latency).
# Other addresses for google.com (not scanned): 2404:6800:4008:802::200e
# rDNS record for 172.217.24.14: tsa01s07-in-f14.1e100.net
# Not shown: 998 filtered ports
# PORT    STATE SERVICE
# 80/tcp  open  http
# 443/tcp open  https
#
# Nmap done: 1 IP address (1 host up) scanned in 3.99 seconds
```

Scan for open ports and OS and version detection (e.g. scan the domain “scanme.nmap.org”)

```
$ nmap -A -T4 scanme.nmap.org
# -A to enable OS and version detection, script scanning, and traceroute; -T4 fo
```

Look up website information (e.g. name server), searches for an object in a RFC 3912 database.

```
whois google.com
```

Show the SSL certificate of a domain

```
openssl s_client -showcerts -connect www.example.com:443
```

Display IP address

```
ip a
```

Display route table

```
ip r
```

Display ARP cache (ARP cache displays the MAC addresses of device in the same network that you have connected to)

```
ip n
```

Add transient IP addres (reset after reboot) (e.g. add 192.168.140.3/24 to device eno16777736)

```
ip address add 192.168.140.3/24 dev eno16777736
```

Persisting network configuration changes

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-enoxxx
# then edit the fields: BOOTPROT, DEVICE, IPADDR, NETMASK, GATEWAY, DNS1 etc
```

Refresh NetworkManager

```
sudo nmcli c reload
```

Restart all interfaces

```
sudo systemctl restart network.service
```

To view hostname, OS, kernal, architecture at the same time!

```
hostnamectl
```

Set hostname (set all transient, static, pretty hostname at once)

```
hostnamectl set-hostname "mynode"
```

Find out the web server (e.g Nginx or Apache) of a website

```
curl -I http://example.com/
# HTTP/1.1 200 OK
# Server: nginx
# Date: Thu, 02 Jan 2020 07:01:07 GMT
# Content-Type: text/html
# Content-Length: 1119
# Connection: keep-alive
# Vary: Accept-Encoding
# Last-Modified: Mon, 09 Sep 2019 10:37:49 GMT
# ETag: "xxxxxx"
# Accept-Ranges: bytes
# Vary: Accept-Encoding
```

Find out the http status code of a URL

```
curl -s -o /dev/null -w "%{http_code}" https://www.google.com
```

Unshorten a shortended URL

```
curl -s -o /dev/null -w "%{redirect_url}" https://bit.ly/34EFwWC
```

Perform network throughput tests

```
# server side:
$ sudo iperf -s -p 80

# client side:
iperf -c <server IP address> --parallel 2 -i 1 -t 2 -p 80
```

To block port 80 (HTTP server) using iptables.

```
sudo iptables -A INPUT -p tcp --dport 80 -j DROP

# only block connection from an IP address
sudo iptables –A INPUT –s <IP> -p tcp –dport 80 –j DROP
```

Data wrangling

[back to top]

Print some words that start with a particular string (e.g. words start with 'phy')

```
# If file is not specified, the file /usr/share/dict/words is used.
look phy|head -n 10
# Phil
# Philadelphia
# Philadelphia's
# Philby
# Philby's
# Philip
# Philippe
# Philippe's
# Philippians
# Philippine
```

Repeat printing string n times (e.g. print 'hello world' five times)

```
printf 'hello world\n%.0s' {1..5}
```

Do not echo the trailing newline

```
username=`echo -n "bashoneliner"`
```

Copy a file to multiple files (e.g copy fileA to file(B-D))

```
tee <fileA fileB fileC fileD >/dev/null
```

Delete all non-printing characters

```
tr -dc '[:print:]' < filename
```

Remove newline / nextline

```
tr --delete '\n' <input.txt >output.txt
```

Replace newline

```
tr '\n' ' ' <filename
```

To uppercase/lowercase

```
tr /a-z/ /A-Z/
```

Translate a range of characters (e.g. substitute a-z into a)

```
echo 'something' |tr a-z a
# aaaaaaaaa
```

Compare two files (e.g. fileA, fileB)

```
diff fileA fileB
# a: added; d:delete; c:changed

# or
sdiff fileA fileB
# side-to-side merge of file differences
```

Compare two files, strip trailing carriage return/ nextline (e.g. fileA, fileB)

```
diff fileA fileB --strip-trailing-cr
```

Number a file (e.g. fileA)

```
nl fileA

#or
nl -nrz fileA
# add leading zeros

#or
nl -w1 -s ' '
# making it simple, blank separate
```

Join two files field by field with tab (default join by the first column of both file, and default separator is space)

```
# fileA and fileB should have the same ordering of lines.
join -t '\t' fileA fileB

# Join using specified field (e.g. column 3 of fileA and column 5 of fileB)
join -1 3 -2 5 fileA fileB
```

Combine/ paste two or more files into columns (e.g. fileA, fileB, fileC)

```
paste fileA fileB fileC
# default tab separate
```

Group/combine rows into one row

```
# e.g.
# AAAA
# BBBB
# CCCC
# DDDD
cat filename|paste - -
# AAAABBBB
# CCCCDDDD
cat filename|paste - - - -
# AAAABBBBCCCCDDDD
```

Fastq to fasta (fastq and fasta are common file formats for bioinformatics sequence data)

```
cat file.fastq | paste - - - - | sed 's/^@/>/g'| cut -f1-2 | tr '\t' '\n' >file
```

Reverse string

```
echo 12345| rev
```

Generate sequence 1-10

```
seq 10
```

Find average of input list/file of integers

```
i=`wc -l filename|cut -d ' ' -f1`; cat filename| echo "scale=2;(`paste -sd+`)/"
```

Generate all combination (e.g. 1,2)

```
echo {1,2}{1,2}
# 1 1, 1 2, 2 1, 2 2
```

Generate all combination (e.g. A,T,C,G)

```
set = {A,T,C,G}
group= 5
for ((i=0; i<$group; i++));do
    repetition=$set$repetition;done
    bash -c "echo "$repetition""
```

Read file content to variable

```
foo=$(<test1)
```

Echo size of variable

```
echo ${#foo}
```

Echo a tab

```
echo -e ' \t '
```

Split file into smaller file

```
# Split by line (e.g. 1000 lines/smallfile)
split -d -l 1000 largefile.txt

# Split by byte without breaking lines across files
split -C 10 largefile.txt
```

Create a large amount of dummy files (e.g 100000 files, 10 bytes each):

```
#1. Create a big file
dd if=/dev/zero of=bigfile bs=1 count=1000000

#2. Split the big file to 100000 10-bytes files
 split -b 10 -a 10 bigfile
```

Rename all files (e.g. remove ABC from all .gz files)

```
rename 's/ABC//' *.gz
```

Remove file extension (e.g remove .gz from filename.gz)

```
basename filename.gz .gz

zcat filename.gz> $(basename filename.gz .gz).unpacked
```

Add file extension to all file(e.g add .txt)

```
rename s/$/.txt/ *
# You can use rename -n s/$/.txt/ * to check the result first, it will only pri
# rename(a, a.txt)
# rename(b, b.txt)
# rename(c, c.txt)
```

Squeeze repeat patterns (e.g. /t/t –> /t)

```
tr -s "/t" < filename
```

Do not print nextline with echo

```
echo -e 'text here \c'
```

View first 50 characters of file

```
head -c 50 file
```

Cut and get last column of a file

```
cat file|rev | cut -d/ -f1 | rev
```

Add one to variable/increment/ i++ a numeric variable (e.g. $var)

```
((var++))
# or
var=$((var+1))
```

Cut the last column

```
cat filename|rev|cut -f1|rev
```

Cat to a file

```
cat >myfile
let me add sth here
exit by control + c
^C
```

Clear the contents of a file (e.g. filename)

```
>filename
```

Append to file (e.g. hihi)

```
echo 'hihi' >>filename
```

Working with json data

```
#install the useful jq package
#sudo apt-get install jq
#e.g. to get all the values of the 'url' key, simply pipe the json to the follo
cat file.json | jq '.url'
```

Decimal to Binary (e.g get binary of 5)

```
D2B=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
echo -e ${D2B[5]}
#00000101
echo -e ${D2B[255]}
#11111111
```

Wrap each input line to fit in specified width (e.g 4 integers per line)

```
echo "00110010101110001101" | fold -w4
# 0011
# 0010
# 1011
# 1000
# 1101
```

Sort a file by column and keep the original order

```
sort -k3,3 -s
```

Right align a column (right align the 2nd column)

```
cat file.txt|rev|column -t|rev
```

To both view and store the output

```
echo 'hihihihi' | tee outputfile.txt
```

```
# use '-a' with tee to append to file.
```

Show non-printing (Ctrl) characters with cat

```
cat -v filename
```

Convert tab to space

```
expand filename
```

Convert space to tab

```
unexpand filename
```

Display file in octal (you can also use od to display hexadecimal, decimal, etc)

```
od filename
```

Reverse cat a file

```
tac filename
```

Reverse the result from `uniq -c`

```
while read a b; do yes $b |head -n $a ;done <test.txt
```

Others

[back to top]

Describe the format and characteristics of image files.

```
identify myimage.png
#myimage.png PNG 1049x747 1049x747+0+0 8-bit sRGB 1.006MB 0.000u 0:00.000
```

Bash auto-complete (e.g. show options “now tomorrow never” when you press’tab’ after typing “dothis”)

More examples

```
complete -W "now tomorrow never" dothis
# ~$ dothis
# never     now       tomorrow
# press 'tab' again to auto-complete after typing 'n' or 't'
```

Displays a calendar

```
# print the current month, today will be highlighted.
cal
# October 2019
# Su Mo Tu We Th Fr Sa
#    1  2  3  4  5
# 6  7  8  9 10 11 12
# 13 14 15 16 17 18 19
# 20 21 22 23 24 25 26
# 27 28 29 30 31

# only display November
cal -m 11
```

Convert the hexadecimal MD5 checksum value into its base64-encoded format.

```
openssl md5 -binary /path/to/file| base64
# NWbeOpeQbtuY0ATWuUeumw==
```

Forces applications to use the default language for output

```
export LC_ALL=C

# to revert:
unset LC_ALL
```

Encode strings as Base64 strings

```
echo test|base64
#dGVzdAo=
```

Get parent directory of current directory

```
dirname `pwd`
```

Read .gz file without extracting

```
zmore filename

# or
zless filename
```

Run command in background, output error file

```
some_commands  &>log &

# or
some_commands 2>log &

# or
some_commands 2>&1| tee logfile

# or
some_commands |& tee logfile

# or
some_commands 2>&1 >>outfile
#0: standard input; 1: standard output; 2: standard error
```

Run multiple commands in background

```
# run sequentially
(sleep 2; sleep 3) &

# run parallelly
sleep 2 & sleep 3 &
```

Run process even when logout (immune to hangups, with output to a non-tty)

```
# e.g. Run myscript.sh even when log out.
nohup bash myscript.sh
```

Send mail

```
echo 'heres the content'| mail -a /path/to/attach_file.txt -s 'mail.subject' me
# use -a flag to set send from (-a "From: some@mail.tld")
```

Convert .xls to csv

```
xls2csv filename
```

Make BEEP sound

```
speaker-test -t sine -f 1000 -l1
```

Set beep duration

```
(speaker-test -t sine -f 1000) & pid=$!;sleep 0.1s;kill -9 $pid
```

Editing your history

```
history -w
vi ~/.bash_history
history -r

#or
history -d [line_number]
```

Interacting with history

```
# list 5 previous command (similar to `history |tail -n 5` but wont print the h
fc -l -5
```

Delete current bash command

```
Ctrl+U

# or
Ctrl+C

# or
```

```
Alt+Shift+#
# to make it to history
```

Add something to history (e.g. “addmetohistory”)

```
# addmetodistory
# just add a "#" before~~
```

Get last history/record filename

```
head !$
```

Clean screen

```
clear
# or simply Ctrl+l
```

Backup with rsync

```
rsync -av filename filename.bak
rsync -av directory directory.bak
rsync -av --ignore_existing directory/ directory.bak
rsync -av --update directory directory.bak

rsync -av directory user@ip_address:/path/to/directory.bak
# skip files that are newer on receiver (i prefer this one!)
```

Make all directories at one time!

```
mkdir -p project/{lib/ext,bin,src,doc/{html,info,pdf},demo/stat}
# -p: make parent directory
# this will create project/doc/html/; project/doc/info; project/lib/ext ,etc
```

Run command only if another command returns zero exit status (well done)

```
cd tmp/ && tar xvf ~/a.tar
```

Run command only if another command returns non-zero exit status (not finish)

```
cd tmp/a/b/c ||mkdir -p tmp/a/b/c
```

Use backslash “\” to break long command

```
cd tmp/a/b/c \
> || \
>mkdir -p tmp/a/b/c
```

List file type of file (e.g. /tmp/)

```
file /tmp/
# tmp/: directory
```

Writing Bash script (‘#!” is called shebang)

```
#!/bin/bash
file=${1#*.}
# remove string before a "."
```

Python simple HTTP Server

```
python -m SimpleHTTPServer
# or when using python3:
python3 -m http.server
```

Read user input

```
read input
echo $input
```

Array

```
declare -a array=()

# or
declare array=()

# or associative array
declare -A array=()
```

Send a directory

```
scp -r directoryname user@ip:/path/to/send
```

Fork bomb

```
# Don't try this at home!
# It is a function that calls itself twice every call until you run out of syste
# A '# ' is added in front for safety reason, remove it when seriously you are
# :(){:|:&};:
```

Use the last argument

```
!$
```

Check last exit code

```
echo $?
```

Extract .xz

```
unxz filename.tar.xz
# then
tar -xf filename.tar
```

Unzip tar.bz2 file (e.g. file.tar.bz2)

```
tar xvfj file.tar.bz2
```

Unzip tar.xz file (e.g. file.tar.xz)

```
unxz file.tar.xz
tar xopf file.tar
```

Extract to a path

```
tar xvf -C /path/to/directory filename.gz
```

Zip the content of a directory without including the directory itself

```
# First cd to the directory, they run:
zip -r -D ../myzipfile .
# you will see the myzipfile.zip in the parent directory (cd ..)
```

Output a y/n repeatedly until killed

```
# 'y':
yes

# or 'n':
yes n

# or 'anything':
yes anything

# pipe yes to other command
yes | rm -r large_directory
```

Create large dummy file of certain size instantly (e.g. 10GiB)

```
fallocate -l 10G 10Gigfile
```

Create dummy file of certain size (e.g. 200mb)

```
dd if=/dev/zero of=//dev/shm/200m bs=1024k count=200
# or
dd if=/dev/zero of=//dev/shm/200m bs=1M count=200

# Standard output:
# 200+0 records in
# 200+0 records out
```

```
# 209715200 bytes (210 MB) copied, 0.0955679 s, 2.2 GB/s
```

Keep /repeatedly executing the same command (e.g Repeat 'wc -l filename' every 1 second)

```
watch -n 1 wc -l filename
```

Print commands and their arguments when execute (e.g. echo `expr 10 + 20`)

```
set -x; echo `expr 10 + 20 `
```

Print some meaningful sentences to you (install fortune first)

```
fortune
```

Colorful (and useful) version of top (install htop first)

```
htop
```

Press any key to continue

```
read -rsp $'Press any key to continue...\n' -n1 key
```

Run sql-like command on files from terminal

```
# download:
# https://github.com/harelba/q
# example:
q -d "," "select c3,c4,c5 from /path/to/file.txt where c3='foo' and c5='boo'"
```

Using Screen for multiple terminal sessions

```
# Create session and attach:
screen

# Create a screen and name it 'test'
screen -S test

# Create detached session foo:
```

```
screen -S foo -d -m

# Detached session foo:
screen: ^a^d

# List sessions:
screen -ls

# Attach last session:
screen -r

# Attach to session foo:
screen -r foo

# Kill session foo:
screen -r foo -X quit


# Scroll:
# Hit your screen prefix combination (C-a / control+A), then hit Escape.
# Move up/down with the arrow keys (↑ and ↓).

# Redirect output of an already running process in Screen:
# (C-a / control+A), then hit 'H'

# Store screen output for Screen:
# Ctrl+A, Shift+H
# You will then find a screen.log file under current directory.
```

Using Tmux for multiple terminal sessions

```
# Create session and attach:
tmux

# Attach to session foo:
tmux attach -t foo

# Detached session foo:
^bd

# List sessions:
tmux ls

# Attach last session:
tmux attach

# Kill session foo:
```

```
tmux kill-session -t foo

# Create detached session foo:
tmux new -s foo -d

# Send command to all panes in tmux:
Ctrl-B
:setw synchronize-panes

# Some tmux pane control commands:
Ctrl-B
#   Panes (splits), Press Ctrl+B, then input the following symbol:
#   %  horizontal split
#   "  vertical split
#   o  swap panes
#   q  show pane numbers
#   x  kill pane
#   space - toggle between layouts

#   Distribute Vertically (rows):
select-layout even-vertical
#   or
Ctrl+b, Alt+2

# Distribute horizontally (columns):
select-layout even-horizontal
#   or
Ctrl+b, Alt+1

# Scroll
Ctrl-b then \[ then you can use your normal navigation keys to scroll around.
Press q to quit scroll mode.
```

Pass password to ssh

```
sshpass -p mypassword ssh root@10.102.14.88 "df -h"
```

Wait for a pid (job) to complete

```
wait %1
# or
wait $PID
wait ${!}
#wait ${!} to wait till the last background process ($! is the PID of the last
```

Convert pdf to txt

```
sudo apt-get install poppler-utils
pdftotext example.pdf example.txt
```

List only directory

```
ls -d */
```

List one file per line.

```
ls -1
# or list all, do not ignore entries starting with .
ls -1a
```

Capture/record/save terminal output (capture everything you type and output)

```
script output.txt
# start using terminal
# to logout the screen session (stop saving the contents), type exit.
```

List contents of directories in a tree-like format.

```
tree
# go to the directory you want to list, and type tree (sudo apt-get install tree
# output:
# home/
# └── project
#     ├── 1
#     ├── 2
#     ├── 3
#     ├── 4
#     └── 5
#

# set level directories deep (e.g. level 1)
tree -L 1
# home/
# └── project
```

Set up virtualenv(sandbox) for python

```
# 1. install virtualenv.
sudo apt-get install virtualenv
# 2. Create a directory (name it .venv or whatever name your want) for your new
virtualenv .venv
# 3. source virtual bin
source .venv/bin/activate
# 4. you can check check if you are now inside a sandbox.
type pip
# 5. Now you can install your pip package, here requirements.txt is simply a tx
pip install -r requirements.txt
# 6. Exit virtual environment
deactivate
```

> More coming!!

Bash-Oneliner is maintained by onceupon.
This page was generated by GitHub Pages.