LeCoupa / **bash-cheatsheet.sh**

Last active yesterday • Report abuse

Embed <script src="https://

Download ZIP

Bash CheatSheet for UNIX Systems --> UPDATED VERSION --> https://github.com/LeCoupa/awesome-cheatsheets

bash-cheatsheet.sh

```
#!/bin/bash
####################################################
# Name: Bash CheatSheet for Mac OSX
#
# A little overlook of the Bash basics
#
# Usage:
#
# Author: J. Le Coupanec
# Date: 2014/11/04
####################################################


# 0. Shortcuts.


CTRL+A  # move to beginning of line
CTRL+B  # moves backward one character
CTRL+C  # halts the current command
CTRL+D  # deletes one character backward or logs out of current session, similar to exit
CTRL+E  # moves to end of line
CTRL+F  # moves forward one character
CTRL+G  # aborts the current editing command and ring the terminal bell
CTRL+J  # same as RETURN
CTRL+K  # deletes (kill) forward to end of line
CTRL+L  # clears screen and redisplay the line
CTRL+M  # same as RETURN
CTRL+N  # next line in command history
CTRL+O  # same as RETURN, then displays next line in history file
CTRL+P  # previous line in command history
CTRL+R  # searches backward
CTRL+S  # searches forward
CTRL+T  # transposes two characters
CTRL+U  # kills backward from point to the beginning of line
CTRL+V  # makes the next character typed verbatim
CTRL+W  # kills the word behind the cursor
CTRL+X  # lists the possible filename completefions of the current word
CTRL+Y  # retrieves (yank) last item killed
CTRL+Z  # stops the current command, resume with fg in the foreground or bg in the background

DELETE  # deletes one character backward
!!      # repeats the last command
exit    # logs out of current session


# 1. Bash Basics.


export              # displays all environment variables

echo $SHELL         # displays the shell you're using
echo $BASH_VERSION  # displays bash version

bash                # if you want to use bash (type exit to go back to your normal shell)
whereis bash        # finds out where bash is on your system

clear               # clears content on window (hide displayed lines)


# 1.1. File Commands.


ls                          # lists your files
```

```
ls -l                       # lists your files in 'long format', which contains the exact size of the file, who owns the file
ls -a                       # lists all files, including hidden files
ln -s <filename> <link>     # creates symbolic link to file
touch <filename>            # creates or updates your file
cat > <filename>            # places standard input into file
more <filename>             # shows the first part of a file (move with space and type q to quit)
head <filename>             # outputs the first 10 lines of file
tail <filename>             # outputs the last 10 lines of file (useful with -f option)
emacs <filename>            # lets you create and edit a file
mv <filename1> <filename2>  # moves a file
cp <filename1> <filename2>  # copies a file
rm <filename>               # removes a file
diff <filename1> <filename2>  # compares files, and shows where they differ
wc <filename>               # tells you how many lines, words and characters there are in a file
chmod -options <filename>   # lets you change the read, write, and execute permissions on your files
gzip <filename>             # compresses files
gunzip <filename>           # uncompresses files compressed by gzip
gzcat <filename>            # lets you look at gzipped file without actually having to gunzip it
lpr <filename>              # print the file
lpq                         # check out the printer queue
lprm <jobnumber>            # remove something from the printer queue
genscript                   # converts plain text files into postscript for printing and gives you some options for formattin
dvips <filename>            # print .dvi files (i.e. files produced by LaTeX)
grep <pattern> <filenames>  # looks for the string in the files
grep -r <pattern> <dir>     # search recursively for pattern in directory


# 1.2. Directory Commands.


mkdir <dirname>  # makes a new directory
cd               # changes to home
cd <dirname>     # changes directory
pwd              # tells you where you currently are


# 1.3. SSH, System Info & Network Commands.


ssh user@host            # connects to host as user
ssh -p <port> user@host  # connects to host on specified port as user
ssh-copy-id user@host    # adds your ssh key to host for user to enable a keyed or passwordless login

whoami                   # returns your username
passwd                   # lets you change your password
quota -v                 # shows what your disk quota is
date                     # shows the current date and time
cal                      # shows the month's calendar
uptime                   # shows current uptime
w                        # displays whois online
finger <user>            # displays information about user
uname -a                 # shows kernel information
man <command>            # shows the manual for specified command
df                       # shows disk usage
du <filename>            # shows the disk usage of the files and directories in filename (du -s give only a total)
last <yourUsername>      # lists your last logins
ps -u yourusername       # lists your processes
kill <PID>               # kills (ends) the processes with the ID you gave
killall <processname>    # kill all processes with the name
top                      # displays your currently active processes
bg                       # lists stopped or background jobs ; resume a stopped job in the background
fg                       # brings the most recent job in the foreground
fg <job>                 # brings job to the foreground

ping <host>              # pings host and outputs results
whois <domain>           # gets whois information for domain
dig <domain>             # gets DNS information for domain
dig -x <host>            # reverses lookup host
wget <file>              # downloads file


# 2. Basic Shell Programming.

```

```
# 2.1. Variables.


varname=value                # defines a variable
varname=value command        # defines a variable to be in the environment of a particular subprocess
echo $varname                # checks a variable's value
echo $$                      # prints process ID of the current shell
echo $!                      # prints process ID of the most recently invoked background job
echo $?                      # displays the exit status of the last command
export VARNAME=value         # defines an environment variable (will be available in subprocesses)

array[0] = val               # several ways to define an array
array[1] = val
array[2] = val
array=([2]=val [0]=val [1]=val)
array(val val val)

${array[i]}                  # displays array's value for this index. If no index is supplied, array element 0 is assumed
${#array[i]}                 # to find out the length of any element in the array
${#array[@]}                 # to find out how many values there are in the array

declare -a                   # the variables are treaded as arrays
declare -f                   # uses funtion names only
declare -F                   # displays function names without definitions
declare -i                   # the variables are treaded as integers
declare -r                   # makes the variables read-only
declare -x                   # marks the variables for export via the environment

${varname:-word}             # if varname exists and isn't null, return its value; otherwise return word
${varname:=word}             # if varname exists and isn't null, return its value; otherwise set it word and then return its va
${varname:?message}          # if varname exists and isn't null, return its value; otherwise print varname, followed by message
${varname:+word}             # if varname exists and isn't null, return word; otherwise return null
${varname:offset:length}     # performs substring expansion. It returns the substring of $varname starting at offset and up to

${variable#pattern}          # if the pattern matches the beginning of the variable's value, delete the shortest part that matc
${variable##pattern}         # if the pattern matches the beginning of the variable's value, delete the longest part that match
${variable%pattern}          # if the pattern matches the end of the variable's value, delete the shortest part that matches an
${variable%%pattern}         # if the pattern matches the end of the variable's value, delete the longest part that matches and
${variable/pattern/string}   # the longest match to pattern in variable is replaced by string. Only the first match is replaced
${variable//pattern/string}  # the longest match to pattern in variable is replaced by string. All matches are replaced

${#varname}                  # returns the length of the value of the variable as a character string

*(patternlist)               # matches zero or more occurences of the given patterns
+(patternlist)               # matches one or more occurences of the given patterns
?(patternlist)               # matches zero or one occurence of the given patterns
@(patternlist)               # matches exactly one of the given patterns
!(patternlist)               # matches anything except one of the given patterns

$(UNIX command)              # command substitution: runs the command and returns standard output


# 2.2. Functions.
# The function refers to passed arguments by position (as if they were positional parameters), that is, $1, $2, and so forth.
# $@ is equal to "$1" "$2"... "$N", where N is the number of positional parameters. $# holds the number of positional parameter


functname() {
  shell commands
}

unset -f functname  # deletes a function definition
declare -f          # displays all defined functions in your login session


# 2.3. Flow Control.


statement1 && statement2  # and operator
statement1 || statement2  # or operator

-a                        # and operator inside a test conditional expression
```

```
-o                      # or operator inside a test conditional expression

str1=str2               # str1 matches str2
str1!=str2              # str1 does not match str2
str1<str2               # str1 is less than str2
str1>str2               # str1 is greater than str2
-n str1                 # str1 is not null (has length greater than 0)
-z str1                 # str1 is null (has length 0)

-a file                 # file exists
-d file                 # file exists and is a directory
-e file                 # file exists; same -a
-f file                 # file exists and is a regular file (i.e., not a directory or other special type of file)
-r file                 # you have read permission
-r file                 # file exists and is not empty
-w file                 # your have write permission
-x file                 # you have execute permission on file, or directory search permission if it is a directory
-N file                 # file was modified since it was last read
-O file                 # you own file
-G file                 # file's group ID matches yours (or one of yours, if you are in multiple groups)
file1 -nt file2         # file1 is newer than file2
file1 -ot file2         # file1 is older than file2

-lt                     # less than
-le                     # less than or equal
-eq                     # equal
-ge                     # greater than or equal
-gt                     # greater than
-ne                     # not equal

if condition
then
  statements
[elif condition
  then statements...]
[else
  statements]
fi

for x := 1 to 10 do
begin
  statements
end

for name [in list]
do
  statements that can use $name
done

for (( initialisation ; ending condition ; update ))
do
  statements...
done

case expression in
  pattern1 )
    statements ;;
  pattern2 )
    statements ;;
  ...
esac

select name [in list]
do
  statements that can use $name
done

while condition; do
  statements
done

until condition; do
  statements
```

```
done


# 3. Command-Line Processing Cycle.


# The default order for command lookup is functions, followed by built-ins, with scripts and executables last.
# There are three built-ins that you can use to override this order: `command`, `builtin` and `enable`.

command  # removes alias and function lookup. Only built-ins and commands found in the search path are executed
builtin  # looks up only built-in commands, ignoring functions and commands found in PATH
enable   # enables and disables shell built-ins

eval     # takes arguments and run them through the command-line processing steps all over again


# 4. Input/Output Redirectors.


cmd1|cmd2  # pipe; takes standard output of cmd1 as standard input to cmd2
> file     # directs standard output to file
< file     # takes standard input from file
>> file    # directs standard output to file; append to file if it already exists
>|file     # forces standard output to file even if noclobber is set
n>|file    # forces output to file from file descriptor n even if noclobber is set
<> file    # uses file as both standard input and standard output
n<>file    # uses file as both input and output for file descriptor n
<<label    # here-document
n>file     # directs file descriptor n to file
n<file     # takes file descriptor n from file
n>>file    # directs file description n to file; append to file if it already exists
n>&        # duplicates standard output to file descriptor n
n<&        # duplicates standard input from file descriptor n
n>&m       # file descriptor n is made to be a copy of the output file descriptor
n<&m       # file descriptor n is made to be a copy of the input file descriptor
&>file     # directs standard output and standard error to file
<&-        # closes the standard input
>&-        # closes the standard output
n>&-       # closes the ouput from file descriptor n
n<&-       # closes the input from file descripor n


# 5. Process Handling.


# To suspend a job, type CTRL+Z while it is running. You can also suspend a job with CTRL+Y.
# This is slightly different from CTRL+Z in that the process is only stopped when it attempts to read input from terminal.
# Of course, to interupt a job, type CTRL+C.

myCommand &  # runs job in the background and prompts back the shell

jobs         # lists all jobs (use with -l to see associated PID)

fg           # brings a background job into the foreground
fg %+        # brings most recently invoked background job
fg %-        # brings second most recently invoked background job
fg %N        # brings job number N
fg %string   # brings job whose command begins with string
fg %?string  # brings job whose command contains string

kill -l      # returns a list of all signals on the system, by name and number
kill PID     # terminates process with specified PID

ps           # prints a line of information about the current running login shell and any processes running under it
ps -a        # selects all processes with a tty except session leaders

trap cmd sig1 sig2  # executes a command when a signal is received by the script
trap "" sig1 sig2   # ignores that signals
trap - sig1 sig2    # resets the action taken when the signal is received to the default

disown <PID|JID>    # removes the process from the list of jobs

wait                # waits until all background jobs have finished
```

```
356
357
358 # 6. Tips and Tricks.
359
360
361 # set an alias
362 cd; nano .bash_profile
363 > alias gentlenode='ssh admin@gentlenode.com -p 3404'  # add your alias in .bash_profile
364
365 # to quickly go to a specific directory
366 cd; nano .bashrc
367 > shopt -s cdable_vars
368 > export websites="/Users/mac/Documents/websites"
369
370 source .bashrc
371 cd websites
372
373
374 # 7. Debugging Shell Programs.
375
376
377 bash -n scriptname  # don't run commands; check for syntax errors only
378 set -o noexec       # alternative (set option in script)
379
380 bash -v scriptname  # echo commands before running them
381 set -o verbose      # alternative (set option in script)
382
383 bash -x scriptname  # echo commands after command-line processing
384 set -o xtrace       # alternative (set option in script)
385
386 trap 'echo $varname' EXIT  # useful when you want to print out the values of variables at the point that your script exits
387
388 function errtrap {
389   es=$?
390   echo "ERROR line $1: Command exited with status $es."
391 }
392
393 trap 'errtrap $LINENO' ERR  # is run whenever a command in the surrounding script or function exists with non-zero status
394
395 function dbgtrap {
396   echo "badvar is $badvar"
397 }
398
399 trap dbgtrap DEBUG  # causes the trap code to be executed before every statement in a function or script
400 # ...section of code in which the problem occurs...
401 trap - DEBUG  # turn off the DEBUG trap
402
403 function returntrap {
404   echo "A return occured"
405 }
406
407 trap returntrap RETURN  # is executed each time a shell function or a script executed with the . or source commands finishes ex
408
```

Yemster79 commented on 24 Nov 2015

Hello I have a CW on Bash Scripting and I do not have a clue one what to do I have no knowledge or experience on Bash Shell Scripting programming

trevordmiller commented on 2 Jan 2016

Wonderful cheat sheet. Thank you for sharing!

cl0482 commented on 16 Feb 2016

Thanks for the cheat sheet!