DEVHINTS.IO

Edit

Bash scripting cheatsheet

Proudly sponsored by

Segment Send data to any tool without having to implement a new API every time.

ethical ad by CodeFund

Example

Variables

String quotes

NAME="John"

```
#!/usr/bin/env bash

NAME="John"
echo "Hello $NAME!"
```

NAME="John"
echo \$NAME
echo "\$NAME"

echo "\${NAME}!"

echo "Hi \$NAME" echo 'Hi \$NAME'

Conditional execution

Functions

Shell execution

```
git commit && git push
git commit || echo "Commit failed"
```

Conditionals

```
if [[ -z "$string" ]]; then
  echo "String is empty"
elif [[ -n "$string" ]]; then
```

get_name() { echo "John" }

```
echo "You are $(get_name)"
```

Strict mode

```
echo "String is not empty"

fi

See: Conditionals

Echo {A,B}.js

{A,B}.

{A,B}.js

{1..5}

See: Brace expansion
```

^t Parameter expansions

Basics	Substitution	Comments
<pre>name="John" echo \${name} echo \${name/J/j} #=> "john" (substitution) echo \${name:0:2} #=> "Jo" (slicing) echo \${name::2} #=> "Jo" (slicing) echo \${name::-1} #=> "Joh" (slicing) echo \${name:(-1)} #=> "n" (slicing from right) echo \${name:(-2):1} #=> "h" (slicing from right) echo \${food:-Cake} #=> \$food or "Cake"</pre>	\${F00%suffix}	# Single line com
	\${F00#prefix}	: '
	\${F00%suffix}	This is a multi line comment
	\${F00##prefix}	
	\${F00/from/to}	
<pre>length=2 echo \${name:0:length} #=> "Jo"</pre>	\${F00//from/to}	Substrings
	\${F00/%from/to}	\${F00:0:3}

Substitution

Racico

Commonto

```
${F00/#from/to}
                                                                                                           ${F00:-3:3}
See: Parameter expansion
                                                    Length
                                                                                                         Default values
STR="/path/to/foo.cpp"
                                                     ${#F00}
echo ${STR%.cpp}
                   # /path/to/foo
                                                                                                           ${F00:-val}
echo ${STR%.cpp}.o # /path/to/foo.o
                                                                                                           ${F00:=val}
echo ${STR##*.}
                   # cpp (extension)
                   # foo.cpp (basepath)
echo ${STR##*/}
                                                                                                           ${F00:+val}
echo ${STR#*/}
                   # path/to/foo.cpp
                                                                                                           ${F00:?message}
echo ${STR##*/}
                   # foo.cpp
                                                                                                           The: is optional (e
echo ${STR/foo/bar} # /path/to/bar.cpp
STR="Hello world"
echo ${STR:6:5} # "world"
echo ${STR:-5:5} # "world"
SRC="/path/to/foo.cpp"
BASE=${SRC##*/} #=> "foo.cpp" (basepath)
DIR=${SRC%$BASE} #=> "/path/to/" (dirpath)
```

^E Loops

With step size Reading lines Forever for i in {5..50.. < file.txt | while read line; do while true; do echo \$line . . . done done

^t Functions

Defining functions

```
myfunc() {
    echo "hello $1"
# Same as above (alternate syntax)
function myfunc() {
    echo "hello $1"
myfunc "John"
```

Returning values

```
myfunc() {
     local myresult='some value'
     echo $myresult
 }
 result="$(myfunc)"
Arguments
```

\$1

```
$#
$*
$@
```

Raising errors

```
myfunc() {
  return 1
if myfunc; then
  echo "success"
else
  echo "failure"
fi
```

See Special parameters.

^t Conditionals

Conditions	File conditions	Example
Note that [[is actually a command/program that returns eigenties, such as grep(1) or ping(1)) can be used as condition,	[[-e FILE]]	if ping -c 1 goog echo "It appear fi
	[[-r FILE]]	
[[-z STRING]]	[[-h FILE]]	if grep -q 'foo'
[[-n STRING]]	[[-d FILE]]	echo "You appea
[[STRING == STRING]]	[[-w FILE]]	11
[[STRING != STRING]]	[[-s FILE]]	<pre># String if [[-z "\$string</pre>
[[NUM -eq NUM]]	[[-f FILE]]	echo "String is elif [[-n "\$stri
[[NUM -ne NUM]]	[[-x FILE]]	echo "String is fi
[[NUM -lt NUM]]	[[FILE1 -nt FILE2]]	
[[NUM -le NUM]]	[[FILE1 -ot FILE2]]	<pre># Combinations if [[X]] && [[</pre>
[[NUM -gt NUM]]	[[FILE1 -ef FILE2]]	 fi
[[NUM -ge NUM]]	Gre	
[[STRING =~ STRING]]		# Equal if [["\$A" == "\$E

```
      (( NUM < NUM ))</td>
      # Regex

      [[ -o noclobber ]]
      if [[ "A" =~ "."

      [[ ! EXPR ]]
      if (( $a < $b ));</td>

      [[ X ]] && [[ Y ]]
      echo "$a is sm

      [[ X ]] || [[ Y ]]
      if [[ -e "file.tx

      echo "file exis
      fi
```

^Ł Arrays

Defining arrays

Fruits=('Apple' 'Banana' 'Orange') Fruits[0]="Apple" Fruits[1]="Banana" Fruits[2]="Orange"

Working with arrays

```
echo ${Fruits[0]}  # Element #0
echo ${Fruits[@]}  # All elements,
echo ${#Fruits[@]}  # Number of elem
echo ${#Fruits}  # String length
echo ${#Fruits[3]}  # String length
echo ${Fruits[@]:3:2}  # Range (from po
```

Operations

Fruits=("\${Fruits[@]}" "Watermelon") # Push Fruits+=('Watermelon') # Also Push

Iteration

```
for i in "${arrayName[@]}"; do echo $i
```

```
Fruits=( ${Fruits[@]/Ap*/} )  # Remove by regex match
unset Fruits[2]  # Remove one item

Fruits=("${Fruits[@]}")  # Duplicate

Fruits=("${Fruits[@]}" "${Veggies[@]}") # Concatenate
lines=(`cat "logfile"`)  # Read from file
```

^t Dictionaries

Iteration Defining Working with dictionaries Iterate over values echo \${sounds[dog]} # Dog's sound declare -A sounds echo \${sounds[@]} # All values for val in "\${sou echo \${!sounds[@]} # All keys echo \$val sounds[dog]="bark" echo \${#sounds[@]} # Number of elements done sounds[cow]="moo" unset sounds[dog] # Delete dog sounds[bird]="tweet" Iterate over keys sounds[wolf]="howl" for key in "\${!so Declares sound as a Dictionary object (aka associative array). echo \$key done

^t Options

Options Glob options

```
set -o noclobber # Avoid overlay files (echo "hi" > foo)
set -o errexit # Used to exit upon error, avoiding cascading errors
set -o pipefail # Unveils hidden failures
set -o nounset # Exposes unset variables
```

```
set -o nullglob  # Non-matching globs are
set -o failglob  # Non-matching globs thro
set -o nocaseglob  # Case insensitive globs
set -o globdots  # Wildcards match dotfile
set -o globstar  # Allow ** for recursive
```

Set GLOBIGNORE as a colon-separated list of pattern

^t History

Commands Expansions

history	!\$
shopt -s histverify	į*
Operations	!-n
Орегалона	!n

11	Execute last command again	
!!:s/ <from>/<t0>/</t0></from>	Replace first occurrence of <from> to <t0> in most recent command</t0></from>	
!!:gs/ <fr0m>/<t0>/</t0></fr0m>	Slices Replace all occurrences of <from> to <t0> in most recent command</t0></from>	
!\$:t	Expand only basenam !!:n	
!\$:h	Expand only director !^	
!! and !\$ can be replaced with any valid expansion.	!\$	
in the state of th		

!!:n-m

!!:n-\$

!! can be replaced with any valid expansion i.e. !c

^t Miscellaneous

Numeric calculations

((a + 200))

```
$((RANDOM%=200)) # Random number 0..200
```

Add 200 to \$a

Inspecting commands

```
command -V cd
#=> "cd is a function/alias/whatever"
```

Trap errors

```
trap 'echo Error at about $LINENO' ERR
```

Subshells

```
(cd somedir; echo "I'm now in $PWD")
pwd # still in first directory
```

Redirection

Case/switch

or

tx

```
traperr() {
 echo "ERROR: ${BASH_SOURCE[1]} at about ${BASH_LINENO[0]}"
set -o errtrace
trap traperr ERR
```

Source relative

```
source "${0%/*}/../share/foo.sh"
```

Directory of script

```
DIR="${0%/*}"
```

Heredoc

```
cat <<END
hello world
END
```

Reading input

```
echo -n "Proceed? [y/n]: "
read ans
echo $ans
```

```
case "$1" in
  start | up)
   vagrant up
   ;;
   echo "Usage: $0 {start|stop|ssh}"
esac
```

printf

```
printf "Hello %s, I'm %s" Sven Olga
#=> "Hello Sven, I'm Olga
```

Getting options

```
while [[ "$1" =~ ^- && ! "$1" == "--" ]]; do
  -V | --version )
```

```
shift; string=$1
```

```
-f | --flag )
```

```
read -n 1 ans # Just one character

Go to previous directory

pwd # /home/user/foo
cd bar/
pwd # /home/user/foo/bar
cd -
pwd # /home/user/foo
```

^t Also see

Bash-hackers wiki (bash-hackers.org)

Shell vars (bash-hackers.org)

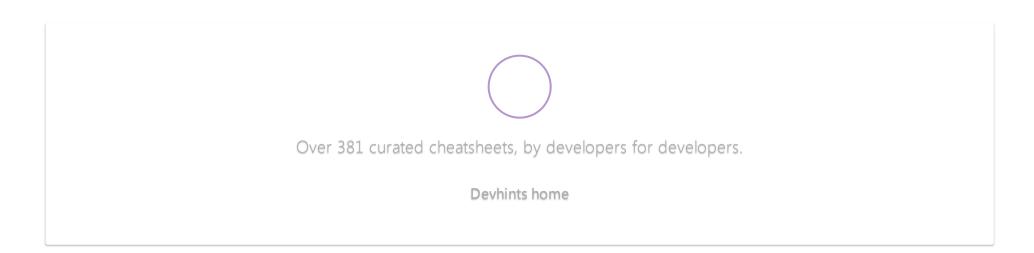
Learn bash in y minutes (learnxinyminutes.com)

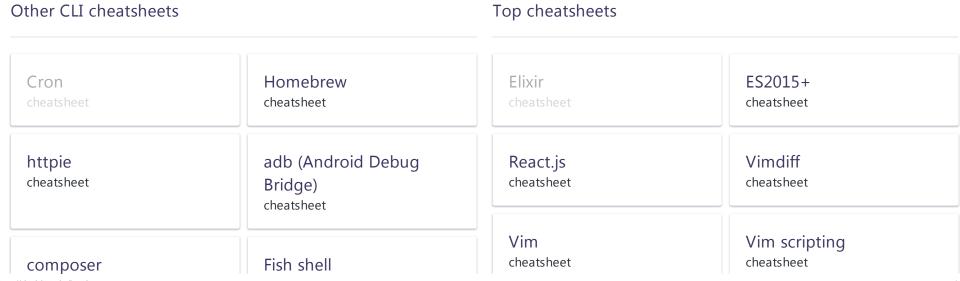
Bash Guide (mywiki.wooledge.org)

ShellCheck (shellcheck.net)

▶ **10 Comments** for this cheatsheet. Write yours!

Search 381+ cheatsheets





cheatsheet

cheatsheet