DEVHINTS.IO

Edit

Bash scripting cheatsheet

Introduction

This is a quick reference to getting started with Bash scripting.

Learn bash in y minutes
(learnxinyminutes.com)

Bash Guide
(mywiki.wooledge.org)

Bash Hackers Wiki
(wiki.bash-hackers.org)

Example

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

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

String quotes

```
name="John"
echo "Hi $name"  #=> Hi J
echo 'Hi $name'  #=> Hi $
```

Shell execution

```
echo "I'm in $(pwd)"
echo "I'm in `pwd`"  # ob
# Same
```

See Command substitution

Conditional execution

```
git commit && git push
git commit || echo "Commi
```

Conditionals

```
if [[ -z "$string" ]]; th
  echo "String is empty"
elif [[ -n "$string" ]];
  echo "String is not emp
fi
```

See: Conditionals

Strict mode

```
set -euo pipefail
IFS=$'\n\t'
```

See: Unofficial bash strict mode

Parameter expansions

Basics

```
name="John"
echo "${name}"
echo "${name/J/j}"    #=>
echo "${name:0:2}"    #=>
echo "${name::2}"     #=>
echo "${name::-1}"    #=>
echo "${name:(-1)}"   #=>
echo "${name:(-2):1}" #=>
echo "${food:-Cake}"  #=>
```

```
length=2
echo "${name:0:length}"
```

See: Parameter expansion

```
str="/path/to/foo.cpp"
echo "${str%.cpp}"    # /
echo "${str%.cpp}.o"  # /
echo "${str%/*}"      # /

echo "${str##*.}"     # c
echo "${str##*/}"     # f

echo "${str#*/}"      # p
echo "${str##*/}"     # f
```

Substitution

`${foo%suffix}`	Remove suffix
`${foo#prefix}`	Remove prefix
`${foo%%suffix}`	Remove long suffix
`${foo/%suffix}`	Re

Manipulation

```
str="HELLO WORLD!"
echo "${str,}"   #=> "hEL
echo "${str,,}"  #=> "hel
echo "${str/foo/bar}" # /
```

```
str="Hello world"
echo "${str:6:5}"   # "wo
echo "${str: -5:5}"  # "w
```

```
src="/path/to/foo.cpp"
base=${src##*/}   #=> "fo
dir=${src%$base}  #=> "/p
```

```
str="hello world!"
echo "${str^}"   #=> "Hel
echo "${str^^}"  #=> "HEL
```

mo ve lon g pre fix

`${foo/#prefix}`	Re mo ve lon g pre fix
`${foo/from/to}`	Rep ch
`${foo//from/to}`	Rep .
`${foo/%from/to}`	Rep lac e suff ix
`${foo/#from/to}`	Rep lac e

# Loops

Basic for loop

```
for i in /etc/rc.*; do
  echo "$i"
done
```

C-like for loop

```
for ((i = 0 ; i < 100 ; i
  echo "$i"
done
```

Reading lines

```
while read -r line; do
  echo "$line"
done <file.txt
```

Forever

```
while true; do
  ···
done
```

# Functions

Defining functions

```
myfunc() {
    echo "hello $1"
}
```

Returning values

```
myfunc() {
    local myresult='some
    echo "$myresult"
```

```
# Same as above (alternat
function myfunc() {
    echo "hello $1"
}
```

```
myfunc "John"
```

```
}
```

```
result=$(myfunc)
```

Arguments

`$#`	Number of arguments
`$*`	All positional arguments (as a single word)
`$@`	All positional arguments (as separate strings)
`$1`	First argument
`$_`	Last argument of the previous command

Note: `$@` and `$*` must be quoted in order to perform as described. Otherwise, they do exactly the same thing (arguments as separate strings)

# Conditionals

Conditions

Note that `[[` is actually a command/program that returns either `0` (true) or `1` (false). Any program that obeys the same logic (like all base utils, such as `grep(1)` or `ping(1)`) can be used as condition, see examples.

File conditions

`[[ -e FILE ]]`	Exists
`[[ -r FILE ]]`	Readabl

`[[ -z STRING ]]`	Empty string
`[[ -n STRING ]]`	Not empty string
`[[ STRING == STRING ]]`	Equal
`[[ STRING != STRING ]]`	Not Eq
`[[ NUM -ne NUM ]]`	N

	e
`[[ -h FILE ]]`	Symlink
`[[ -d FILE ]]`	Directory
`[[ -w FILE ]]`	Writable
`[[ -s FILE ]]`	Size is > 0 bytes
`[[ -f FILE ]]`	File
`[[ -x FILE ]]`	e

# Arrays

Defining arrays

```
Fruits=('Apple' 'Banana' 'Orange')
```

```
Fruits[0]="Apple"
Fruits[1]="Banana"
Fruits[2]="Orange"
```

Operations

Working

```
echo "$
echo "$
echo "$
echo "$
echo "$
echo "$
echo "$
echo "$
```

```
Fruits=("${Fruits[@]}" "Watermelon")    #
Fruits+=('Watermelon')                  #
Fruits=( "${Fruits[@]/Ap*/}" )          #
unset Fruits[2]                         #
Fruits=("${Fruits[@]}")                 #
Fruits=("${Fruits[@]}" "${Veggies[@]}") #
lines=(`cat "logfile"`)                 #
```

e s s t h a n

# Dictionaries

Defining

`[[ NUM -le NUM ]]` L e s s t h a n o r e q u al

```
declare -A sounds
```

```
sounds[dog]="bark"
sounds[cow]="moo"
sounds[bird]="tweet"
sounds[wolf]="howl"
```

Declares sound as a Dictionary object (aka associative array).

`[[ NUM -gt NUM ]]` G r e a t

# Options

## Options

```
set -o noclobber  # Avoid overlay files (
set -o errexit    # Used to exit upon err
set -o pipefail   # Unveils hidden failur
set -o nounset    # Exposes unset variabl
```

n

`[[ FILE1 -nt FILE2 ]]` 1 is m or e re c e nt th a n 2

Iteration

```
for i i
  echo
done
```

Working with dictionaries

`[[ FILE1 -ot FILE2 ]]` 2 is m or e re c e nt th a n 1

```
echo "${sounds[dog]}" # D
echo "${sounds[@]}"   # A
echo "${!sounds[@]}"  # A
echo "${#sounds[@]}"  # N
unset sounds[dog]     # D
```

`[[ FILE1 -ef FILE2 ]]` S a m e fil e s

Glob op

```
shopt -
shopt -
shopt -
shopt -
shopt -
```

`[[ NUM -ge NUM ]]`	Greater than

Set `GLOB` patterns

History

Commands

`history`	Show history
`shopt -s histverify`	Don't execute expanded result immediately

Operations

`!!`	Execute last command again
`!!:s/<FROM>/<TO>/`	Replace first occurrence of `<FROM>` to `<TO>` in most recent command
`!!:gs/<FROM>/<TO>/`	Replace all occurrences of `<FROM>` to `<TO>` in most recent command
`!$:t`	Expand only basename from last parameter of most recent command
`!$:h`	Expand only directory from last parameter of most

Expansi

`!$`
`!*`
`!-n`
`!n`

Slices

`!!:n`
`!^`
`!$`
`!!:n-m`
`!!:n-$`
`!!` can b

Miscellaneous

recent command

Numeric calculations

```
$((a + 200))      # Add 200 to $a
```

```
$(($RANDOM%200))  # Random number 0..199
```

```
declare -i count  # Declare as type integ
count+=1          # Increment
```

E
is
e
n

Inspecting commands

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

d

Trap errors

```
[[ ! EXPR ]]
```

N

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

or

```
traperr() {
  echo "ERROR: ${BASH_SOURCE[1]} at about
}

set -o errtrace
trap traperr ERR
```

Source relative

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

Transform strings

`-c`	Operations apply to

i.e. !cat

Subshel

```
(cd som
pwd # s
```

Redirect

```
python
python
python
python
python
python
python
echo "$
```

```
python
diff <(
```

Case/sw

```
case "$
  start
    vag
    ;;

  *)
    ech
    ;;
esac
```

printf

```
printf
#=> "He

printf
#=> "1
```

	characters not in the given set
`-d`	Delete characters
`-s`	Replaces repeated characters with single occurrence
`-t`	Truncates
`[:upper:]`	All upper case letters
`[:lower:]`	All lower case letters
`[:digit:]`	All digits
`[:space:]`	All whitespace
`[:alpha:]`	All letters
`[:alnum:]`	All letters and digits

Example

Heredoc

```
echo "Welcome To Devhints" | tr '[:lower:

cat <<END
hello world
END
```

Special variables

`$?`	Exit status of last task
`$!`	PID of last background task
`$$`	PID of shell
`$0`	Filename of the shell script

```
printf
#=> "Th

printf
# forma
printf
```

Director

```
dir=${0
```

Getting

```
while [
  -V |
    ech
    exi
    ;;
  -s |
    shi
    ;;
  -f |
    fla
    ;;
esac; s
if [[ "
```

Reading

```
echo -n
read -r
echo "$
```

The `-r` c
behavio

```
read -n
```

`$_`	Last argument of the previous command
`${PIPESTATUS[n]}`	return value of piped commands (array)

Check for command's result

```
if ping -c 1 google.com; then
  echo "It appears you have a working int
fi
```

Go to pr

```
pwd # /
cd bar/
pwd # /
cd -
pwd # /
```

Grep ch

```
if grep
  echo
fi
```

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)

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

devhints.io / Search 358+ cheatsheets

Over 358 curated cheatsheets, by developers for developers.

Devhints home

Other CLI cheatsheets

Cron cheatsheet

Homebrew cheatsheet

httpie cheatsheet

adb (Android Debug Bridge) cheatsheet

composer cheatsheet

Fish shell cheatsheet

Top cheatsheets

Elixir cheatsheet

ES2015+ cheatsheet

React.js cheatsheet

Vimdiff cheatsheet

Vim cheatsheet

Vim scripting cheatsheet