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)

String quotes

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

Shell execution

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

See Command substitution

Functions

```
get_name() {
  echo "John"
}
```

Example

```
#!/usr/
name="J
echo "H
```

Variable

```
name="J
echo $n
"$
"$
```

```
ral
```

```
wildcar
options
sop
```

Conditic

```
git com
git com
```

Conditic

```
if [[ -
```

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

See: Unofficial bash strict mode

```
echo {A
```

```
{A,B}
```

```
{A,B}.j
```

```
{1..5}
```

```
{{1..3}
```

See: Bra

Parameter expansions

Basics

```
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"
```

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

See: Parameter expansion

```
str="/path/to/foo.cpp"
```

Substitu

```
${foo%s
```

```
${foo#p
```

```
${foo%%
```

```
${foo/%
```

```
${foo##
```

```
${foo/#
```

```
${foo/f
```

```
${foo//
```

```
${foo/%
```

`${foo:-val}`	`$foo`, or `val` if unset (or null)
`${foo:=val}`	Set `$foo` to `val` if unset (or null)
`${foo:+val}`	`val` if `$foo` is set (and not null)
`${foo:?message}`	Show error message and exit if `$foo` is unset (or null)

Omitting the `:` removes the (non)nullity checks, e.g. `${foo-val}` expands to `val` if unset otherwise `$foo`.

```
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)
```

Loops

Basic for loop

C-like fo

```
for i in {1..5}; do
    echo "Welcome $i"
done
```

With step size

```
for i in {5..50..5}; do
    echo "Welcome $i"
done
```

```
while t
  ...
done
```

Functions

Defining functions

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

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

```
myfunc "John"
```

Returni

```
myfunc(
    loc
    ech
}
```

```
result=
```

Raising

```
myfunc(
  retur
}
```

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).

See Special parameters.

Conditionals

Conditions

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

`[[ -z STRING ]]`	Empty
`[[ -n STRING ]]`	Not empty
`[[ STRING == STRING ]]`	
`[[ STRING != STRING ]]`	Not
`[[ NUM -eq NUM ]]`	
`[[ NUM -ne NUM ]]`	Not
`[[ NUM -lt NUM ]]`	Less
`[[ NUM -le NUM ]]`	Less than or
`[[ NUM -gt NUM ]]`	Greater than
`[[ NUM -ge NUM ]]`	Greater than or equal
`[[ STRING =~ STRING ]]`	Regexp
`(( NUM < NUM ))`	Numeric cond

More conditions

`[[ -o noclobber ]]`	If OPTIONNAME is en

File con

```
[[ -e F
[[ -r F
[[ -h F
[[ -d F
[[ -w F
[[ -s F
[[ -f F
[[ -x F
[[ FILE
[[ FILE
[[ FILE
```

Example

```
# Strin
if [[ -
  echo
elif [[
  echo
else
```

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

```
[[ X && Y ]]
```

```
[[ X || Y ]]
```

```
  echo
fi
```

```
# Combi
if [[ X
  ...
fi
```

```
# Equal
if [[ "
```

```
# Regex
if [[ "
```

```
if (( $
   echo
fi
```

```
if [[ -
  echo
fi
```

Arrays

Defining arrays

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

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

Working

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

Operations

```
Fruits=("${Fruits[@]}" "Watermelon")    # Push
Fruits+=('Watermelon')                  # Also Push
Fruits=( "${Fruits[@]/Ap*/}" )          # Remove by regex match
unset Fruits[2]                         # Remove one item
Fruits=("${Fruits[@]}")                 # Duplicate
Fruits=("${Fruits[@]}" "${Veggies[@]}") # Concatenate
```

Iteratior

```
for i i
  echo
done
```

```
lines=(`cat "logfile"`)                # Read from file
```

Dictionaries

Defining

```
declare -A sounds
```

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

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

Working

```
echo "$
echo "$
echo "$
echo "$
unset s
```

Iteratior

Iterate ov

```
for val
  echo
done
```

Iterate ov

```
for key
  echo
done
```

Options

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
```

Glob op

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

Set GLOB

History

Commands

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

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
`!$:t`	Expand only basename from last parameter of most recent com
`!$:h`	Expand only directory from last parameter of most recent com

`!!` and `!$` can be replaced with any valid expansion.

Expansi

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

Slices

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

`!!` can b

Miscellaneous

Numeric calculations

```
$((a + 200))      # Add 200 to $a
$(($RANDOM%200))  # Random number 0..199
declare -i count  # Declare as type integer
```

Subshel

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

Redirec

```
if grep -q 'foo' ~/.bash_history; then
  echo "You appear to have typed 'foo' in the past"
fi
```

```
pwd # /home/user/foo
```

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

[:lower:]	All lower case le
[:digit:]	All
[:space:]	All whitespace
[:alpha:]	All letters
[:alnum:]	All letters and digits

Example

```
echo "Welcome To Devhints" | tr '[:lower:]' '[:upper:]'
WELCOME TO DEVHINTS
```

$0

$_

${PIPES

See Spe

♯ 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)

composer
cheatsheet

Fish shell
cheatsheet

cheatsheet

cheatsheet