

Bash Cheatsheet

This document contains `bash` specific commands / syntax which may not be completely POSIX complaint.

POSIX complaint shell scripting doc will be ready after this one ;)

Variables

- Define

```
1 name='PJ'
2 age=10
3 os=$(uname)
```

- Use

```
1 echo $name
2 printf '- %s\n' \
3         $age \
4         $os
```

- Execute

```
$ bash tmp.sh
```

- or if it has shebang and execute permission

```
$ ./tmp.sh
PJ
- 10
- Linux
$_
```

- Length

```
1 a='Hello World'
2 b=1917
3 echo ${#a} # 11
4 echo ${#b} # 4
```

Special Variables

Variable	Description
\$0	Name of script
\$1, \$2, \$3, ...	First, Second, Third, ... argument of script
\$#	Number of arguments were passed to the script
\$@	All arguments of the script (can be iterated)
\$*	All arguments of the script (cannot be iterated)
\$?	Return value of the last execution in script
\$\$	The PID of the script
\$USER	The user which is running the script (username)
\$HOSTNAME	The host name of the machine
\$LINENO	Current line number inside script
\$RANDOM	Random number

Input

- Basic

```
1 read input_variable
2 echo ${input_variable}
```

- with prompt message

```
1 read -p 'are you sure? ' input_variable
```

- silent input

```
1 read -sp 'Input is silent: ' input_variable
```

Arrays

- Define

```
1 files=('f1.txt' 'f2.txt' 'f3.txt')
2 echo ${files[0]} # the first element
3 echo ${files[*]} # all elements
4 echo ${files[@]} # same
5 echo ${#files} # size/length of array
```

- Add elements

```
1 files+=('f4.txt')
```

- Remove elements

```
1 unset files[0]
```

Arithmetic

Basic Expressions

Operator	Operation
+	Addition
-	Subtraction
*	Multiply
/	Deviation
%	Remainder
var++	Increase by 1
var--	Decrease by 1

- let

```

1 let a=4+5 # 9
2 let 'A = 4 + 6' # 10
3 let a++ # 10
4 let A-- # 9

```

- **expr**

```

1 var_two=$( expr 4 \* 5 ) # 20

```

- **Double Parentheses**

```

1 a=$((3 + 5)) # 8
2 b=$(( a + 3 )) # 11
3 (( b++ )) # 12

```

Conditions

- **Test Operations**

Operator	Operation
!	Not

- **String Operations**

Operator	Operation
-z	Is null
-n	Is not null
==	Is equal
!=	Is not equal

- **Numerical Operations**

Operator	Operation
-eq	equal
-lt	less than
-gt	greater than
-le	less-equal to

Operator	Operation
-ge	greater-equal to

- File Operations

Operator	Operation
-e	Exists
-d	Exists and it's a directory
-f	Exists and it's a file
-r	Exists and has <i>read</i> permission
-w	Exists and has <i>write</i> permission
-x	Exists and has <i>execute</i> permission
-s	Exists and it's not empty

if

- Basic

```

1 if [[ `echo $(date +%s) % 5 | bc` -eq 0 ]]; then
2     echo "It can be divided by 5 without any reminder"
3 elif [[ ${second_condition} ]]; then
4     echo "The second is true"
5 else
6     printf '%s\n' \
7         "Nothing is true" \
8         "Everything is permitted"
9 fi

```

- Nested

```

1 if [ 10 -gt 5 ]; then
2     echo True
3     if (( 10 % 2 == 0 )); then
4         echo and Even
5     fi
6 fi

```

- In-line

```
1  [[ ${some_condition} ]] && echo "it's true" || echo 'false'
```

- Boolean Operation

```
1  if [[ -r $1 ]] && [[ -s $1 ]]; then
2      echo "$1 has read permission and contains some data"
3  fi
```

case

- Basic

```
1  case $TERM in
2      xterm)
3          PS1="\u@\h \w]\$ "
4          ;;
5      xterm-256colors)
6          PS1="\[\033[01;34m]\u@\h\[\033[00m\]:\[\033[01;32m]\W\[\033[00m]\$ "
7          ;;
8      *)
9          PS1='$ '
10         ;;
11 esac
```

- (Un)capitalized

```
1  case $1 in
2      -[hH])
3          help
4          ;;
5      -[vV])
6          version
7          ;;
8      -[yY])
9          choice=true
10         ;;
11 esac
```

Loops

- for

```
1 for i in {1..10}
2 do
3     echo ${i}
4 done
```

- while

```
1 counter=1
2 while [[ ${counter} -le 9 ]]; do
3     echo "${counter}"
4     ((counter++))
5 done
```

- until

```
1 counter=1
2 until [[ ${counter} -gt 9 ]]
3 do
4     echo "${counter}"
5     ((counter++))
6 done
```

- select

```
1 names='Kyle Cartman Stan Quit'
2 PS3='Select character: '
3 select name in ${names}; do
4     if [[ $name == 'Quit' ]]; then
5         break
6     fi
7     echo Hello ${name}
8 done
9 echo Bye
```

Controlling Loops: break and continue

- break

```
1  for value in $1/*; do
2      used=$(df $1 | tail -1 | awk '{print $5;}' | sed 's/%//')
3      if [[ ${used} -gt 90 ]]; then
4          echo Low disk space > /dev/stderr
5          break
6      fi
7      cp $value $1/backup/
8  done
```

Functions

- Define

```
1  # first
2  name() {
3      <commands>
4  }
5  # second
6  function name() {
7      <commands>
8  }
```

- Call the function

```
1  print_something() {
2      echo "Hello World!"
3  }
4  print_something
```

- Passing arguments

```
1  say_hello_to() {
2      echo "Hello ${1}!"
3  }
```



```
4 say_hello_to Mars
```

- *return* value

```
1 return_something() {  
2     return ${RANDOM}  
3 }  
4 return_something  
5 echo "The previous function returned $?"
```

- Variable scope

```
1 the_variable='This is global'  
2 echo "Outside of function: ${the_variable}"  
3 change_var() {  
4     local the_variable='This is local'  
5     echo "Inside of function: ${the_variable}"  
6 }  
7 echo "Outside of function: ${the_variable}"  
8 change_var  
9 echo "Outside of function: ${the_variable}"
```

- Overwriting commands

```
1 ls() {  
2     command ls -lhgX  
3 }  
4 ls
```

