

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

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

- Use

```
echo $name
printf '- %s\n' \
    $age \
    $os
```

- Execute

```
$ bash tmp.sh
```

- or if it has shebang and execute permission

```
$ ./tmp.sh
```

```
PJ
```

```
- 10
```

```
- Linux
```

`$_`

- Length

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

Special Variables

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

Input

- Basic

```
read input_variable
echo ${input_variable}
```

- with prompt message

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

- silent input

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

Arrays

- Define

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

- Add elements

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

- Remove elements

```
unset files[0]
```

Arithmetic

Basic Expressions

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

- let

```

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

```

- **expr**

```

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

```

- **Double Parentheses**

```

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

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

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

- Nested

```

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

```

- In-line

```

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

```

- Boolean Operation

```

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

```

case

- Basic

```

case $TERM in
    xterm)
        PS1="\u@\h \w]\$ "
        ;;
    xterm-256colors)
        PS1="\[\033[01;34m]\u@\h:\[\033[01;32m]\W\[\033[00m]\$ "
        ;;
    *)
        PS1='$ '
        ;;
esac

```

- (Un)capitalized

```

case $1 in
    -[hH])
        help
    ;;
    -[vV])
        version
    ;;
    -[yY])
        choice=true
    ;;
esac

```

Loops

- **for**

```

for i in {1..10}
do
    echo ${i}
done

```

- **while**

```

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

```

- **until**

```

counter=1
until [[ ${counter} -gt 9 ]]
do
    echo "${counter}"
    ((counter++))
done

```

- select

```

names='Kyle Cartman Stan Quit'
PS3='Select character: '
select name in ${names}; do
    if [[ $name == 'Quit' ]]; then
        break
    fi
    echo Hello ${name}
done
echo Bye

```

Controlling Loops: break and continue

- break

```

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

```

Functions

- Define


```
# first
name() {
    <commands>
}

# second
function name() {
    <commands>
}
```

- Call the function

```
print_something() {
    echo "Hello World!"
}

print_something
```

- Passing arguments

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

say_hello_to Mars
```

- *return* value

```
return_something() {
    return ${RANDOM}
}

return_something
echo "The previous function returned $?"
```

- Variable scope

```
the_variable='This is global'
echo "Outside of function: ${the_variable}"
change_var() {
    local the_variable='This is local'
    echo "Inside of function: ${the_variable}"
}
echo "Outside of function: ${the_variable}"
change_var
echo "Outside of function: ${the_variable}"
```

- Overwriting commands

```
ls() {
    command ls -lhgX
}
ls
```