Bash Commands

Help

```
<command> --help
```

Get command help

```
man <command>
```

Get man page of command

type: `/<search term>` to search
=> type: `n` go to next match
=> type: `N` go to prev match

Basics

```
who
```

Show all logged in users

```
whoami
```

Show my username

```
ls -la
```

list all files and folders - also hidden (a)

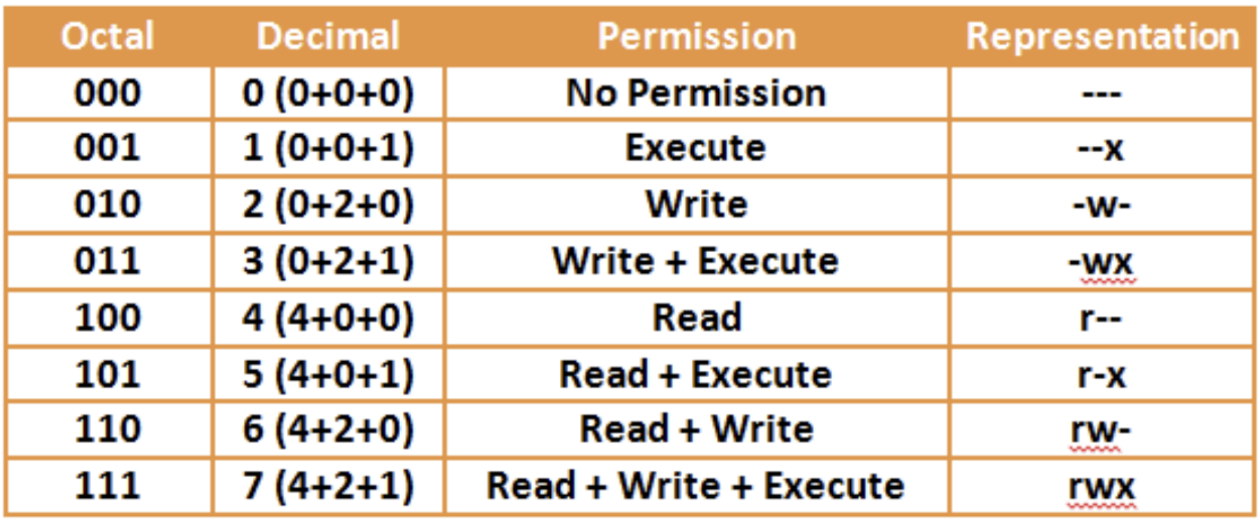
```
-rwxrwx--x  1 vmadmin vmadmin  460 Aug  5  2018 verzweigung7.sh
```

rwxrwx--x =

rwx	rwx	--x
User	Group	Others

```
chmod 777 file.txt
```

set permission for file.txt



Octal	Decimal	Permission	Representation
000	0 (0+0+0)	No Permission	---
001	1 (0+0+1)	Execute	--x
010	2 (0+2+0)	Write	-w-
011	3 (0+2+1)	Write + Execute	-wx
100	4 (4+0+0)	Read	r--
101	5 (4+0+1)	Read + Execute	r-x
110	6 (4+2+0)	Read + Write	rw-
111	7 (4+2+1)	Read + Write + Execute	rwx

```
sudo nautilus
```

gnu file manager as root

```
pwd
```

Get current directory

```
find / -name "*usb*.conf" 2>/dev/null
```

find from root any file matching
suppress all stderr

```
cut -d ' ' -f 2,4 blumenartikel.txt
```

get column 2 & 4 (`-f`) from file using delimiter (space) (`-d`)

```
sort -n -k 2 tmp.out
```

Sort the 2nd col (`-k`) ASC using number (`-n`) from file tmp.out

---

```
ln -s /home/vmadmin/M122
ln -s /home/vmadmin/M122 /home/vmadmin/Desktop/M122_Alias # with target
specified (M122_Alias)
```

Create symbolic link

```
ls -la | tee dir.txt # shows console output
ls -la >dir.txt # won't show console output
```

Show stdout in console and write to file

```
which ifconfig # out: /usr/sbin/ifconfig
```

Show path of executable

Vars

```
HELLO="World"
echo $HELLO
```

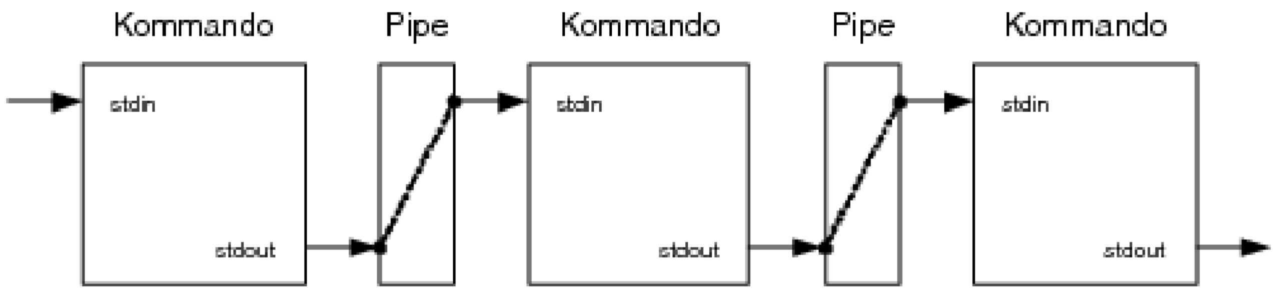
- `$PATH` *Locations containing executables*
- `$?` *Exit status of last command / script*

Parameters

```
./script.sh hello world
```

- `$0` = `./script.sh` - script path
- `$@` = `hello world` - array of all arguments
- `$1` = `hello`
- `$2` = `world`
- ...
- `$#` = `2` - number of arguments

Pipe |



```
cat outputblumen.txt | more -1
```

Get content of file and pass to more to show only 1 line

Pre exec $()

```
chmod 777 $(find /home/vmadmin/ -name *.txt)
```

Change permission for every file in /home/vmadmin with filetype `.txt`

Write to file

```
<command> 2>/dev/null                 # hide all stderr in console

# into file:
<command> >no_errors.txt              # stdout
<command> 2>errors.txt                # stderr
<command> &>out_and_errors.txt        # stdout & stderr

<command> >stdout.txt 2>stderr.txt    # stdout & stderr - separate file
```

Read input

```
read -p "Enter smth: " # limit number of chars using `-N <int: chars>`
echo $USER_INPUT
```

if

```
if [[ $1 -eq 'hello' && $1 =~ ^[Hh].*[^\.]$ ]] # see Comparer below
then
    echo 'world'
elif [[ $1 = 'hallo' ]]
then
    echo 'welt'
else
    echo 'else'
fi

# using function and return code
function my_func {
    true
}
function my_func_false {
    false
}

if $(my_func) && [[ 12 =~ [0-9]+ ]] # func and regex combined
then
    echo "true"
fi

if ! $(my_func_false)
then
    echo "false"
fi
```

Comparer	Explanation
`! <EXPRESSION>`	The EXPRESSION is false.
`-n <STRING>`	The length of STRING is greater than zero.
`-z <STRING>`	The lengh of STRING is zero (ie it is empty).
`<STRING1> = <STRING2>`	STRING1 is equal to STRING2

Comparer	Explanation
`<STRING1> != <STRING2>`	STRING1 is not equal to STRING2
`<STRING> =~ <EXPRESSION>`	STRING matches expression => `expression without quotes`
`<INTEGER1> -eq <INTEGER2>`	INTEGER1 is numerically equal to INTEGER2
`<INTEGER1> -gt <INTEGER2>`	INTEGER1 is numerically greater than INTEGER2
`<INTEGER1> -lt <INTEGER2>`	INTEGER1 is numerically less than INTEGER2
`-d <FILE>`	FILE exists and is a directory.
`-e <FILE>`	FILE exists.
`-r <FILE>`	FILE exists and the read permission is granted.
`-s <FILE>`	FILE exists and it's size is greater than zero (ie. it is not empty).
`-w <FILE>`	FILE exists and the write permission is granted.
`-x <FILE>`	FILE exists and the execute permission is granted.

switch case

```
echo -n "Enter the name of a country: "
read COUNTRY

case $COUNTRY in
  "Switzerland") # string
    echo -n "Schweiz"
    ;;

  [0-9]) # using regex
    echo -n "any Number"
    ;;

  *)
    echo -n "unknown"
    ;;
```

```
esac
```

while

```
exit=0

while [ $exit -ne 1 ]
do

  i=0
  while [ $i -lt 10 ]
  do
    echo "."
    sleep 1
    ((i++))
  done

  $exit=1
done
```

for

```
for i in {0..3} # for item in [LIST]
do
  echo "Number: $i"
done

# declaring array:
BOOKS=('In Search of Lost Time' 'Don Quixote' 'Ulysses' 'The Great
Gatsby')
for book in "${BOOKS[@]}"; do
  echo "Book: $book"
done

# using i:
for ((i = 0 ; i <= 1000 ; i++)); do
  echo "Counter: $i"
done

# using files:
for file in /home/vmadmin/Desktop/M122; do
    echo $file
done
```

functions

```
function hello_world {
  echo "hello world"
}

globalVar="change me"

hello_user () {
  # param: $1 - firstname
  # param: $2 - lastname
  local localVar=10
  globalVar="changed"

  echo "hello $1 $2"

  return 0 # only numeric - else use global vars
}

# call function -> without `()`
hello_world
hello_user "john" "doe"
# access return value
echo $?
```

Access to parameters is the same as for whole scripts.
→ Parameters

```
#region Functions

function is_num {
    # string - string to test
    REGEX='^[0-9]+$'
    if [[ $1 =~ $REGEX ]]; then
        true
    else
        false
    fi
}

function is_decimal {
    # string - string to test
    REGEX='^[0-9]+(\.[0-9]+$|$)'
    if [[ $1 =~ $REGEX ]]; then
        true
    else
        false
    fi
```

```
}

function is_str {
    # string - string to test
    REGEX='[A-Za-z0-9_]+'
    if [[ $1 =~ $REGEX ]]; then
        true
    else
        false
    fi
}

function file_exists {
    # file
    if [[ -f "$1" && -n "$1" ]]; then
        true
    else
        false
    fi
}

function folder_exists {
    # folder
    if [[ -d "$1" && -n "$1" ]]; then
        true
    else
        false
    fi
}

function files_are_equal {
    # file - file 1
    # file - file 2
    diff $1 $2 &>/dev/null
    return $?
}

function file_contains {
    # file
    # expression - regex expression
    grep -e $2 $1 &>/dev/null
    return $?
}

function folder_empty {
    # folder - foldername
    if [[ -z "$(ls $1 2>/dev/null)" ]]; then
        true
    else
```

```
        false
    fi
}

function folder_contains {
    # folder
    # expression - or filename
    find $1 -printf "%f\n" | grep -e $2 &>/dev/null
    return $?
}

function kill_process {
    # string - process name
    kill $(ps -Alf | grep "$1" | tr -s [:blank:] '\t' | head -n1 | cut -
f4) &>/dev/null
    return $?
}

function get_file_line {
    # file
    # line - line number
    sed "$2q;d" $1
}

USER_INPUT=""
function get_input {
    # string - prompt to show
    echo -n "$1: "
    read
    USER_INPUT=$REPLY
}

function read_confirmation {
    read -p "Do you want to continue? [Y/n] " -n 1 -r
    REGEX="^[Yy]$"
    if [[ $REPLY =~ $REGEX ]]; then
        true
    else
        false
    fi
}

function set_var {
    # string - var name
    # string - var value
    printf -v "$1" "%s" "$2"
}

function get_filename {
    # string - full file path
    basename $1
}

function validate_param {
    # string - var name
    # string - var value
    if [[ $1 =~ _dir$ ]]; then
        folder_exists $2
        return $?
    elif [[ $1 =~ _file$ ]]; then
        file_exists $2
        return $?
    elif [[ $1 =~ _num$ || $1 =~ _int$ ]]; then
        is_num $2
        return $?
    elif [[ $1 =~ _decimal$ || $1 =~ _dec$ ]]; then
        is_dec $2
        return $?
    elif [[ $1 =~ _string$ || $1 =~ _str$ ]]; then
        is_str $2
        return $?
    fi
}

function validate_params {
    for i in ${!PARAMS[@]}; do
        param=${PARAMS[$i]}
        value=${!param}
        if ! $(validate_param $param $value); then
            print_usage
            script_error
        fi
    done
}

function script_error {
    exit 1
}

function script_success {
    exit 0
}

function print_usage {
    echo "usage: $0 ${PARAMS[*]}" # string / num / decimal / dir / file
}

#endregion Functions
```

Command Master Table

Command	Purpose
`basename`	strip directory and suffix from filenames
`cat`	concatenate files and print on the standard output
`cd`	change directory
`chgrp`	change file group ownership
`chmod`	change file permissions
`chown`	change file owner and group
`cp`	copy files and directories
`cut`	remove sections from each line of files
`date`	print or set the system date and time
`dc`	an arbitrary precision calculator
`echo`	display a line of text
`exit`	terminate script and return exit-code
`find`	search for files in a directory hierarchy
`gedit`	text editor for the GNOME Desktop
`grep`	print lines that match patterns
`head`	output the first part of files
`ifconfig`	configure a network interface
`kill`	send a signal to a process
`ls`	list directory contents
`man`	an interface to the system reference manuals
`mkdir`	make directories

Command	Purpose
`mv`	move (rename) files
`nano`	Nano's ANOther editor, inspired by Pico
`nautilus`	a file manager for GNOME
`ps`	report a snapshot of the current processes.
`pwd`	print name of current/working directory
`read`	get user input
`rmdir`	remove empty directories
`rm`	remove files or directories
`sort`	sort lines of text files
`sudo`	execute a command as another user
`tail`	output the last part of files
`tee`	read from standard input and write to standard output and files
`top`	display Linux processes
`touch`	create file without content
`tr`	translate or delete characters
`wc`	print word count for file
`which`	locate a command
`whoami`	print effective userid
`who`	show who is logged on

Source: `man`

Regex

> *Reference:* `122 DossierL.pdf` *- Page 59*

```
# grep '<regex expression>' <file>
grep '1$' mrolympia.dat
```

Create New Script

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

# Programm: HalloWelt.sh
# Version: 1.0
# Autor: TKluser
# Datum: 08.09.2021
# Lizenz: MIT
# Beschreibung: <TODO>

echo "hello world"
```

First line is called **Shebang**.
There are two options of defining:

- `#!/bin/bash` direct path to interpreter
- `#!/usr/bin/env bash` get the path to interpreter from env

Summary & Appendix

- 01
- 02

```
PARAMS=("folder1_dir" "folder2_dir")
INTERACTIVE=true
if [[ $# -ne ${#PARAMS[@]} ]]; then
    if [[ "$INTERACTIVE" = "true" ]]; then
        # for every param ask for user input
        for i in ${!PARAMS[@]}; do
            PARAM_NAME="${PARAMS[$i]}"
            get_input ${PARAMS[$i]}
            set_var $PARAM_NAME $USER_INPUT
        done
    else
        echo "error: not all paramters specified"
        print_usage
        script_error
    fi
```

```
else
    # set all params to corresponding param variable
    for i in ${!PARAMS[@]}; do
        PARAM_NAME="${PARAMS[$i]}"
        index=$((i + 1))
        VALUE=${!index}
        set_var $PARAM_NAME $VALUE
    done
fi

# Parameter validation based on type
validate_params

# Further Parameter validation
if $(folder_empty $folder2_dir); then
    echo "error: folder2_dir cannot be empty"
    script_error
fi

#region Main Code

if $(folder_contains _test/ "log.log"); then
    echo "folder contains"
fi

if $(file_contains hallo.txt ".*ha.*"); then
    echo "file contains"
fi

#endregion Main Code

# Script End
read -p "Press any key to continue ..." -n 1 -t 10
script_success
```