BASH

COLLABORATORS

	TITLE : BASH		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bas Terwijn	2018	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
	2018		BT

Contents

1 Introductie

- Bash Programming Language
  - imperatieve programmeertaal
  - shell scripting (quick-and-dirty)
  - domein specifieke taal
- Hello World:

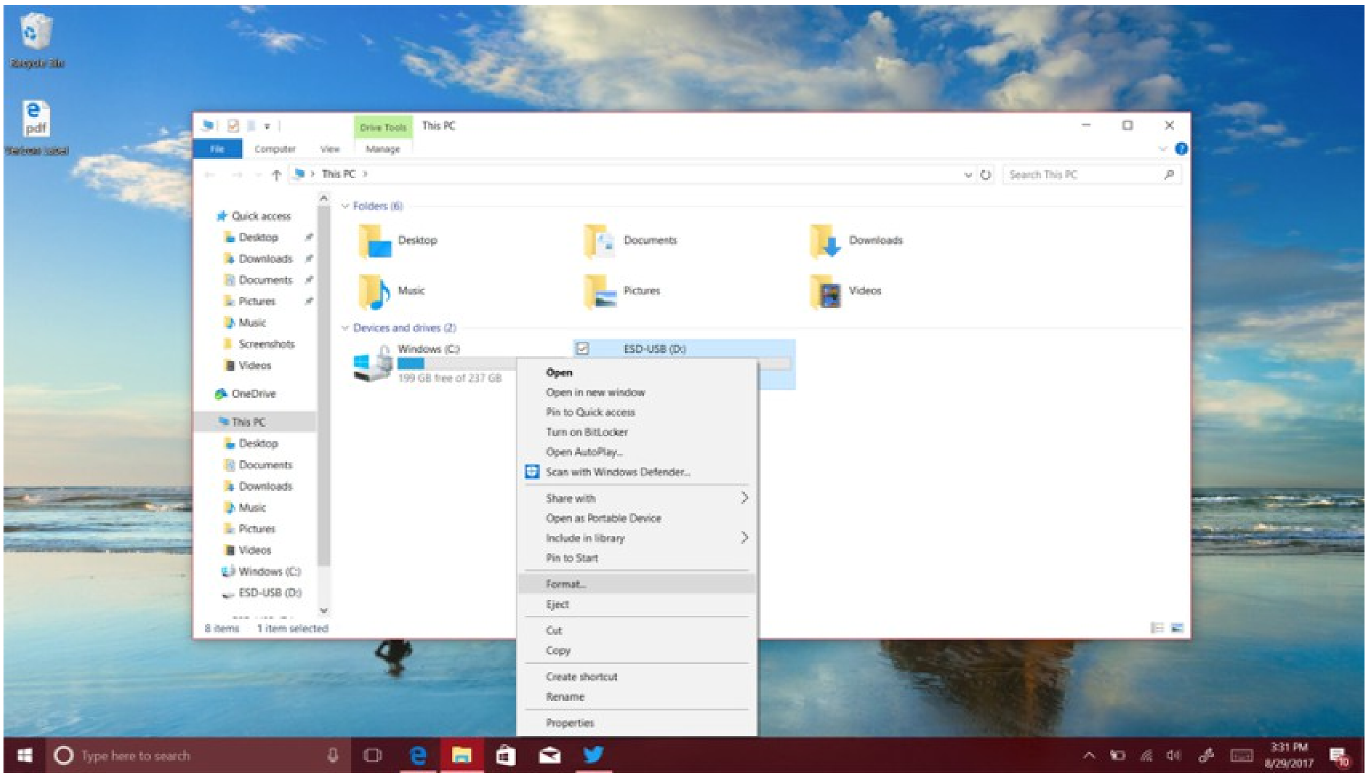
```
$ echo "Hello World!"
Hello World!
```

- Bash alternatieven:
  - sh, csh, tsh, and ksh
  - ash, dash, zsh, and fish

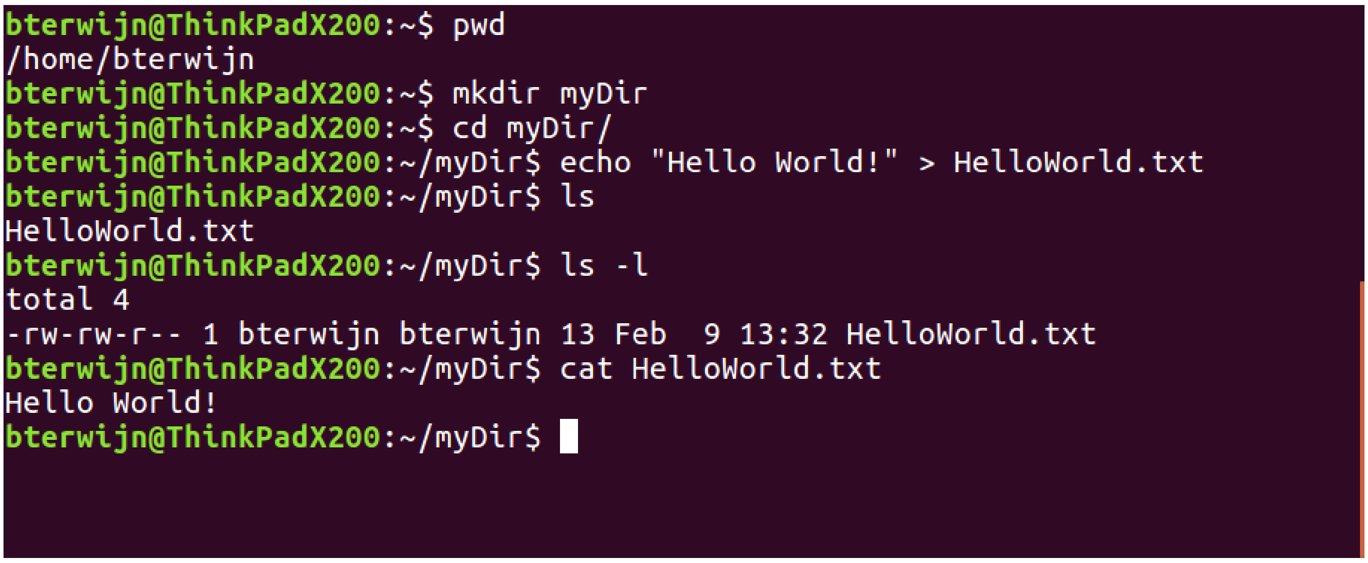
2 Inhoud

- Human Computer Interaction (shell)
- Bash
  - execute
  - variables
  - control flow
  - functions
  - file I/O
  - string manipulation
- Opgave

3 Human Computer Interaction



4 Human Computer Interaction



```
bterwijn@ThinkPadX200:~$ pwd
/home/bterwijn
bterwijn@ThinkPadX200:~$ mkdir myDir
bterwijn@ThinkPadX200:~$ cd myDir/
bterwijn@ThinkPadX200:~/myDir$ echo "Hello World!" > HelloWorld.txt
bterwijn@ThinkPadX200:~/myDir$ ls
HelloWorld.txt
bterwijn@ThinkPadX200:~/myDir$ ls -l
total 4
-rw-rw-r-- 1 bterwijn bterwijn 13 Feb  9 13:32 HelloWorld.txt
bterwijn@ThinkPadX200:~/myDir$ cat HelloWorld.txt
Hello World!
bterwijn@ThinkPadX200:~/myDir$
```

Shell commando’s zijn executables in *$PATH* directories

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which ls
/bin/ls
```

5 Shell shortcuts

https://www.howtogeek.com/howto/ubuntu/keyboard-shortcuts-for-bash-command-shell-for-ubuntu-debian-suse-redhat-linux-etc/-

- *Tab* — Name completion
- *Ctrl+P*, *arrow up* — history walk back
- *Ctrl+N*, *arrow down* — history walk forward
- *Ctrl+R* — search history
- *Ctrl+A* — Home
- *Ctrl+E* — End
- *Alt+F* — forward word
- *Alt+B* — back word
- *Alt+D* — delete word
- *Alt+.* — last word in previous command

6 Shell Commands: Filesystem

- `mv` — move (rename) files
- `cp` — copy files
- `rm` — remove files
- `rmdir` — remove empty directory
- `ln` — make links between files
- `touch` — change file timestamps (create empty files)
- `chmod` — change file/directory mode bits (read, write, execute/search)
- `basename` — extract file name from path
- `dirname` — extract dirctory name from path

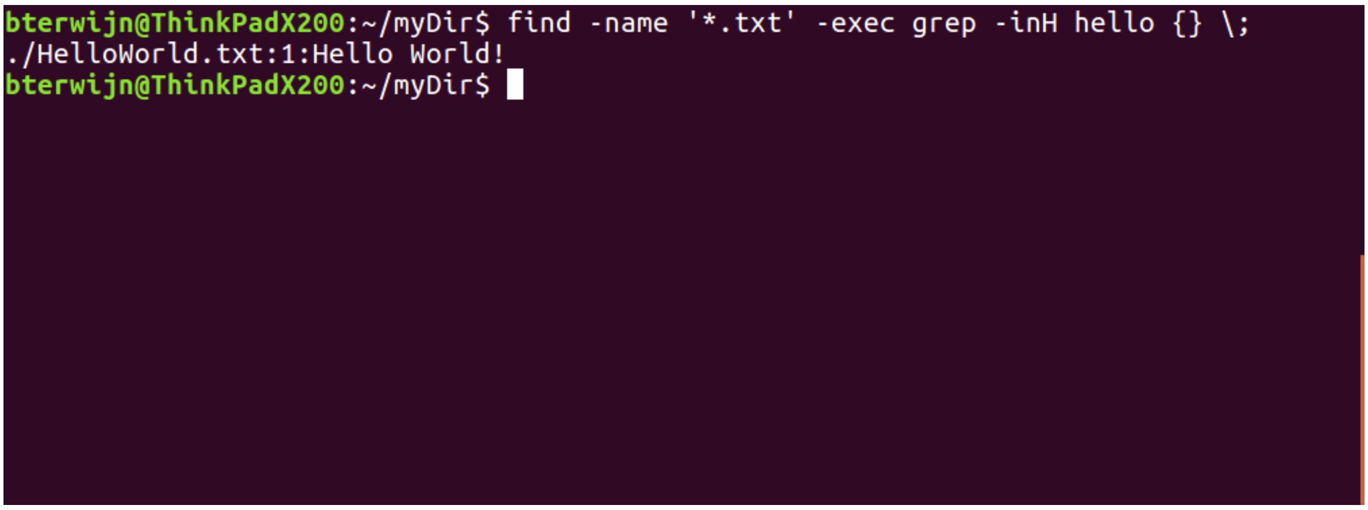
7 Shell Commands: File Contents

- `more` — file *pager* with forward movement
- `less` — file *pager* with both forward and backward movement
- `head` — output the first lines of files
- `tail` — output the last lines of files
- `sort` — alphabetically sort lines of files
- `wc` — count lines, words, and characters in files

8 More Shell Commands

- `grep` — output lines matching a pattern
- `tr` — translate (or delete) characters
- `sed` — stream editor for filtering and transforming text
- `diff` — find line based differences between files
- `find` — search for files by name/type/owner/size/etc.
- `man` — manual page for each command

9 Search for "hello"



```
bterwijn@ThinkPadX200:~/myDir$ find -name '*.txt' -exec grep -inH hello {} \;
./HelloWorld.txt:1:Hello World!
bterwijn@ThinkPadX200:~/myDir$
```

10 Want more complex things?

- Use Bash scripts to combine commands

```
$ for i in $(find -name '*.txt')
$ do
$   grep -inH hello $i
$ done
./HelloWorld.txt:1:Hello World!
```

11 Bash Resources

- Advanced Bash-Scripting Guide
- http://wiki.bash-hackers.org/doku.php
- http://mywiki.wooledge.org/BashGuide
- Bash Reference Manual

12 Bash execute

12.1 Interactive

```
$ # A ’#’ starts a comment
$ echo "Hello World!"
Hello World!
```

12.2 Sourcing

hello_world.src:

```
# This file can be sourced from Bash
echo Hello World!
```

```
$ source hello_world.src
Hello World!
```

13 Bash execute

13.1 Scripting

shebang.sh:

```
#!/bin/bash
echo "arguments: $1 $2 $3 (nr: $#)"
```

```
$ chmod +x shebang.sh
$ ./shebang.sh a b c
arguments: a b c (nr: 3)
```

14 Variables

It is easiest to think of variables in Bash as stored strings

```
$ # You will almost always need quoting
$ MSG="Hello World!"
$ echo $MSG
Hello World!
```

Field splitting makes quoting important

```
$ MSG="foo    bar"
$ echo $MSG "," "$MSG" "," ’$MSG’
foo bar , foo    bar , $MSG
```

Variables are not restricted to arguments

```
$ CMD=echo
$ $CMD Unbelievable
Unbelievable
```

15 Variables: Parameter Substitution

```
$ echo "${X}, ${X-bar}, ${X:-baz}"
, bar, baz
```

```
$ X=
$ echo "${X}, ${X-bar}, ${X:-baz}"
, , baz
```

```
$ X="foo"
$ echo "${X}, ${X-bar}, ${X:-baz}"
foo, foo, foo
```

```
$ unset X
$ echo "${X}, ${X-bar}, ${X:-baz}"
, bar, baz
```

16 Control Flow

if statement

```
$ answer="42"
$ if [[ "$answer" = "42" ]]; then
$   echo "expression evaluated as true"
$ else
$   echo "expression evaluated as false"
$ fi
expression evaluated as true
```

alternatively

```
$ [[ "$answer" = "42" ]] && echo "expression evaluated as true"
expression evaluated as true
```

17 Control Flow: Case

```
argument_count() {
  case $# in
   0)
    echo "No arguments"
    ;;
   1)
    echo "One argument"
    ;;
   2|3)
    echo "A few arguments"
    ;;
   *)
    echo "$# arguments"
  esac
}
```

18 Conditions

strings: [["a" = "b"]]

- = — equal
- *!=* — not equal
- < — smaller alphabetically
- > — larger alphabetically
- *-n* — not empty
- *-z* — empty

numbers: [["1" -lt "2"]]

- *-lt* — <, less than
- *-gt* — >, greater than
- *-le* — ⇐, less than or equal
- *-ge* — >=, greater than or equal
- *-eq* — ==, equal
- *-ne* — !=, not equal

19 Conditions

```
$ true && echo "Yes!"
Yes!
$ false && echo "Yes!"
$ false || echo "No!"
No!
$ true && echo "Yes!" || echo "Give me more"
Yes!
```

Note
These constructs are lazy and left associative.

20 Control Flow

for loop over list

```
for i in $( ls )
do
  echo "item: $i"
done
```

for loop over range

```
for i in $(seq 1 10); do
  echo -n "$i "
done
```

```
1 2 3 4 5 6 7 8 9 10
```

21 Control Flow

while loop

```
COUNTER=0
while [[  $COUNTER -lt 10 ]]; do
  echo The counter is $COUNTER
  let COUNTER+=1
done
```

use *break* and *continue* for additional loop control

22 Functions

```
function myFunction {
  echo "nr arguments: $#"
  add=$(( $1 + $2 ))
  subtract=$(( $1 - $2 ))
}

function print_add {
  echo $add
}

myFunction "3" "4"
echo $add
echo $subtract
print_add
```

```
nr arguments: 2
7
-1
7
```

23 Functions: Variable Scope

```
X="X_outside"
Y="Y_outside"
myFunction() {
  local X
  X="X_inside"
  Y="Y_inside"
  echo "X:$X  Y:$Y"
}
```

```
$ myFunction
X:X_inside  Y:Y_inside
$ echo "X:$X  Y:$Y"
X:X_outside  Y:Y_inside
```

24 Functions: Return value

```
function divide {
  if [[ "$#" -ge "2" ]] && [[ "$2" -ne "0" ]]; then
    divide_result=$(( $1 / $2 ))
    return 1  # state: good
  else
    return 0  # state: error
  fi
}

divide 5 0
if [[ "$?" != "0" ]] ; then echo $divide_result; fi
divide 5 2
if [[ "$?" != "0" ]] ; then echo $divide_result; fi

exit 0
```

```
2
```

25 File I/O

```
echo "overwrite file" > myFile.txt
echo "append line"   >> myFile.txt
echo "append line"   >> myFile.txt
```

readStdIn.sh:

```
while read -r line; do
    echo "read line: $line"
done
```

```
$ source readStdIn.sh < myFile.txt
read line: overwrite file
read line: append line
read line: append line
```

```
filename="myFile.txt"
while read -r line; do
    echo "read line: $line"
done < "$filename"
```

26 Pipes

char_replace.sh:

```
#!/bin/bash
IFS="" # no word splitting
while read -n 1 -d '\0' i; do
    if [[ "$i" = "$1" ]]; then
        echo -n "$2"
    else
        echo -n $i;
    fi
done < "${3:-/dev/stdin}"
```

```
$ echo "This is a test" | ./char_replace.sh ' ' '_'
This_is_a_test
$ echo "This is a test" > test.txt
$ ./char_replace.sh ' ' '_' test.txt
This_is_a_test
$ ./char_replace.sh ' ' '_' test.txt | ./char_replace.sh '_' ' ' | ./char_replace. ↩
    sh 't' 'X'
This is a XesX
```

27 String manipulation

$ X="aabbccaabbcc"		
$ echo "length: ${#X}"	length: 12	length
$ echo "${X:0:1} , ${X:4}, ${X:(-2)}"	a , ccaabbcc, cc	cut
$ echo "${X/a/ZZ} , ${X//a/ZZ}"	ZZabbccaabbcc , ZZZZbbccZZZZbbcc	replace
$ echo "${X#a*b} , ${X##a*b}"	bccaabbcc , cc	delete front
$ echo "${X%b*c} , ${X%%b*c}"	aabbccaab , aa	delete back

28 String manipulation

shopt -s extglob # extended globbing

```
?(pattern-list) Matches zero or one occurrence of the given patterns
*(pattern-list) Matches zero or more occurrences of the given patterns
+(pattern-list) Matches one or more occurrences of the given patterns
@(pattern-list) Matches one of the given patterns
!(pattern-list) Matches anything except one of the given patterns
```

pattern-list is a list of one or more patterns separated by a |.

```
$ X="aabbccaabbcc"
$ shopt -s extglob
$ echo "$\{X##?(a|b)\} , $\{X##+(a|b)\}"
```

abbccaabbcc , ccaabbcc

29 Regular Expression

a	matches literal character
[abc]	matches any character given
[a-z]	matches any character in range
.	matches any single character
?	matches preceding item at most once
*	matches preceding item zero or more times
+	matches preceding item one or more times
ˆ	matches beginning of a line
$	matches end of a line
()	capture group

30 Regular Expression

```
$ X="aabbccaabbcc"
$ if [[ "$X" =~ ^([ab]*)([ac]*)([bc]*)$ ]]; then
$   echo "${BASH_REMATCH[1]} , ${BASH_REMATCH[2]} , ${BASH_REMATCH[3]}"
$ fi
aabb , ccaa , bbcc
```

```
$if [[ "$X" =~ ([a-z]*)a ]]; then
$   echo "${BASH_REMATCH[1]}"
$fi
aabbcca
```

31 Opgave