Operating System fundamentals

Scripting part 2

Karel de Grote Hogeschool

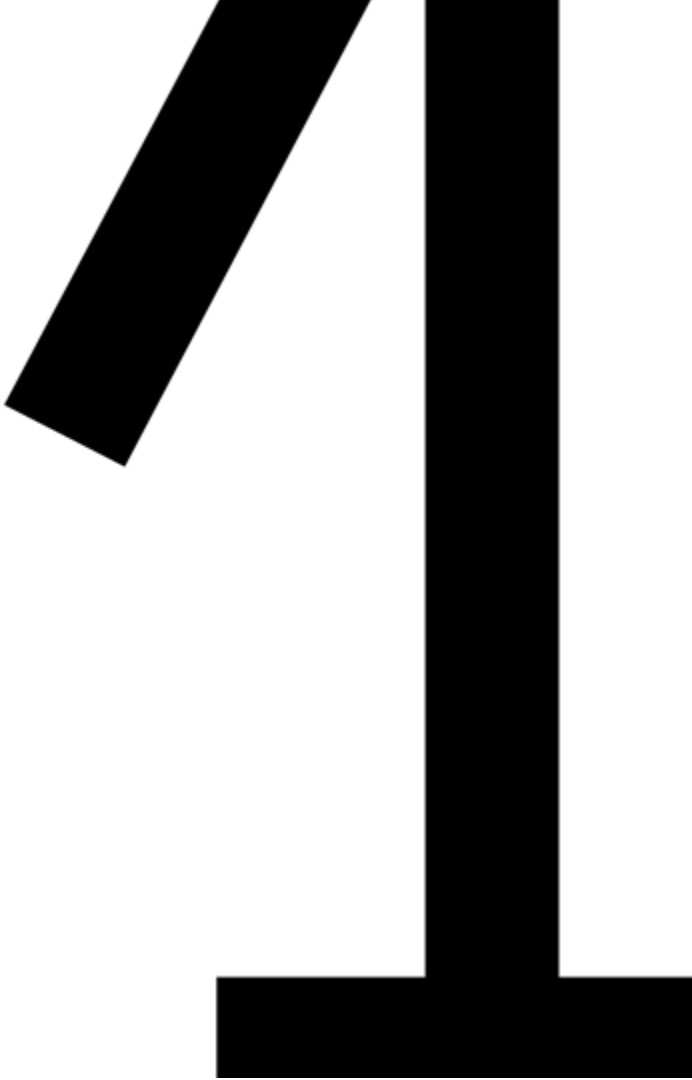
Inhoud

1. exit codes
2. conditions
3. iterations
4. switch-case
5. functions

Course text

- Not in RedHat course!
- Chapter 9 Scripting, Part 2
  - Exit Codes
  - Conditions
  - Iterations
  - Functions

Exit codes



Exit codes

- every program has an "exit code"
  - is the return-value of main()
  - in java: System.exit(exit_code);
  - in script: "exit 0"
- a program ending without an error
  --> exit-code = 0
- a program resulting in an error
  --> exit-code > 0

Exit codes

you can request the last exit code using "${?}"

```
ls
echo ${?}
find / -name "blablah" 2>/dev/null
echo ${?}
sgddfg
echo ${?}
true
echo ${?}
false
echo ${?}
mkdir test ; echo ${?}
```

Exit codes

- commands can be executed depending on the exit status of another command:
  - comm1 **&&** comm2
    - execute comm2 only if comm1 succeeds
    - example: mkdir t **&&** echo "dir created"
  - comm1 **||** comm2
    - execute comm2 only if comm1 fails
    - example: mkdir t **||** echo "dir already existed"

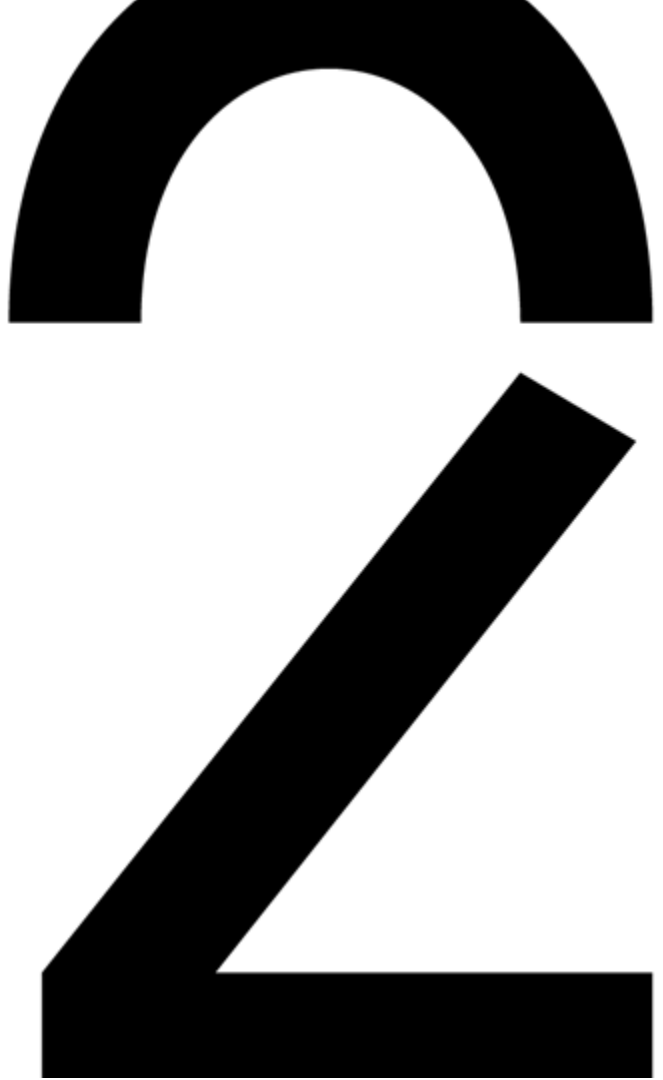
Application in script

```
#!/bin/bash

today=$(date +%Y%m%d)
backup_dir=/tmp
backup_file=backup${today}.tgz
source_dir="/root"
logfile=/var/log/backup.log

touch /root/test 2>/dev/null || { echo "Execute as root!" 1>&2; exit 1; }
echo "Backing up to ${backup_file}..."
tar -zcf "${backup_dir}/${backup_file}" "${source_dir}" "${@}" 2>/dev/null
sync
echo "${today}: backup successful" >> ${logfile}
echo "Done."
```

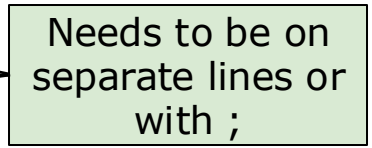
Conditions



If … then … else … fi

- uses the exit code of a command as a boolean
- voorbeeld:

```
if cp source dest 2>/dev/null
then
 echo "success"
else
 echo "failure"
fi
```




```
if cp source dest 2>/dev/null; then echo "success"; else echo "failure"; fi
```

Three ways to write an “if”

```
if ...
then
  ...
fi
```

```
if ...
then
  ...
else
  ...
fi
```

```
if ...
then
  ...
elif ...
then
  ...
else
  ...
fi
```

Example: what does this script do?

```
#!/bin/bash
if grep -F "${1}" /etc/passwd 2>/dev/null 1>/dev/null
then
 echo "user ${1} exists"
elif grep -iF "${1}" /etc/passwd 2>/dev/null 1>/dev/null
then
 echo "user ${1} almost exists..."
else
 echo "user ${1} does not exist"
fi
```

The test command

- “test” will test a certain condition and returns a boolean as exit status (0=true)

```
test -f /etc/passwd
echo ${?}
test "${USER}" = "john"
echo ${?}
test 5 -gt 7
echo ${?}
```

Test: other syntax

```
if test "${USER}" != "john"
then
 echo "not equal to john"
else
 echo "equal to john"
fi
```

```
if [ "${USER}" != "john" ]
then
 echo "not equal to john"
else
 echo "equal to john"
fi
```

[is a command! Try `which [`

KdG
Karel de Grote Hogeschool

Test files

see man page or
https://en.wikipedia.org/wiki/Test_(Unix)

- [-f filename]: test if filename is a file
- [-d filename]: test if filename is a directory
- [-e filename]: test if filename exists
- [-r file]: test if file is readable
- [-w file]: test if file is writable
- [-x file]: test if file is executable
- [file1 -nt file2]: test if file1 is newer than file2
- [file1 -ot file2]: test if file1 is older than file2

Test strings

- [string1 = string2]: test if string1 is equal to string2
- [string1 != string2]: test if string1 is not equal to string2
- [-n string]: test if string is not empty
- [-z string]: test if string is empty

Test integers

- [number1 **-lt** number2] : number1 < number2
- [number1 **-le** number2] : number1 <= number2
- [number1 **-eq** number2] : number1 == number2
- [number1 **-gt** number2] : number1 > number2
- [number1 **-ge** number2] : number1 >= number2
- [number1 **-ne** number2] : number1 != number2

Test: boolean operators

- [**!** ${1} -eq 5] : not-operator
- [${1} -eq 5 **-a** ${2} -eq 6] : AND
- [${1} -eq 5 **-o** ${2} -eq 6] : OR
- [${1} -eq 5] **&&** [${2} -eq 6] : AND
- [${1} -eq 5] **||** [${2} -eq 6] : OR
- [${1} -eq 5 **-a \(** ${2} -eq 6 **-o** ${3} -eq 7 **\)**]: brackets

Application in backup script

```
#!/bin/bash

today=$(date +%Y%m%d)
backup_dir=/tmp
backup_file=backup${today}.tgz
source_dir="/root"
logfile=/var/log/backup.log

[ $(id -u) -eq 0 ] || { echo "Execute as root!" 1>&2; exit 1; }
echo "Backing up to ${backup_file}..."
tar -zcf "${backup_dir}/${backup_file}" "${source_dir}" "${@}" \
2>/dev/null
sync
echo "${today}: backup successful" >>${logfile}
echo "Done."
```

KdG
Karel de Grote Hogeschool

Application in backup script

```
#!/bin/bash

today=$(date +%Y%m%d)
backup_dir=/tmp
backup_file=backup${today}.tgz
source_dir="/root"
logfile=/var/log/backup.log

if [ $(id -u) -ne 0 ] ; then
 echo "Execute as root!" 1>&2
 exit 1
fi
echo "Backing up to ${backup_file}..."
tar -zcf "${backup_dir}/${backup_file}" "${source_dir}" "${@}" \
2>/dev/null
sync
echo "${today}: backup successful" >>${logfile}
echo "Done."
```

Exercise

- write a script "copy2dir" that expects 2 arguments
  - if there are not exactly 2 arguments an error is shown and the script ends with exit code 1
  - if the second argument is not a directory an error is shown and the script ends with exit code 2
  - if the first argument is not a file or not readable the script shows and error and ends with exit code 3
  - if all arguments are valid, the file is copied to the directory
- make your script executable and test it

Iterations

For-loop

```
#!/bin/bash
for file in file1 /tmp/file2 /file3
do
  [ -x "${file}" ] && echo "${file} is executable"
done
```

For-loop

```
#!/bin/bash
for file in $(ls)
do
  [ -x "${file}" ] && echo "${file} is executable"
done
```

For-loop

```
#!/bin/bash
for file in /tmp/*
do
   [ -x "${file}" ] && echo "${file} is executable"
done
```

While

example:

```
#!/bin/bash
while [ ${#} -ne 0 ]
do
  echo "${1}"
  shift
done
```

Iterating through all lines of a file

```
#!/bin/bash
while read -r line
do
  echo "${line}"
done < file
```

Until

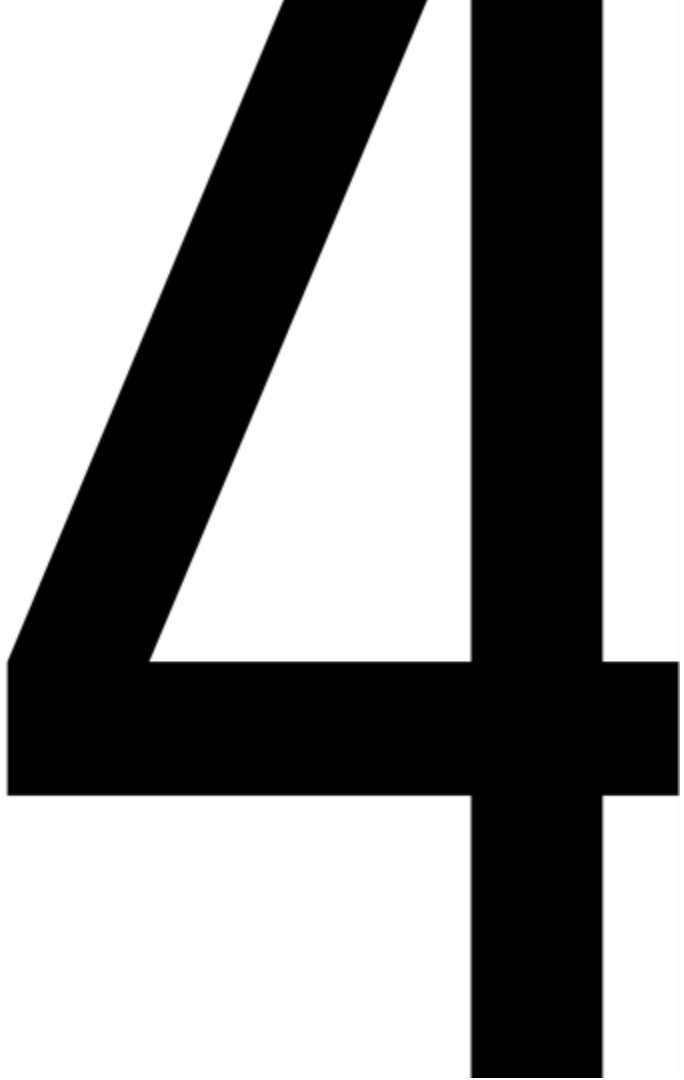
```
#!/bin/bash
counter=20
until [ "${counter}" -lt 10 ]
do
 echo counter "${counter}"
 counter=$((counter-1))
done
```

Exercise

- Write a script that loops through all files and directories in the current directory and states for each if it is a file or a directory.

- Expand the previous script such that an optional argument can specify the directory to be used (instead of the current one) Check if the parameter is a directory.

switch-case



switch-case

```
#!/bin/bash
case ${1} in
  start) echo "starting service" ;;
  stop) echo "stopping service" ;;
esac
```

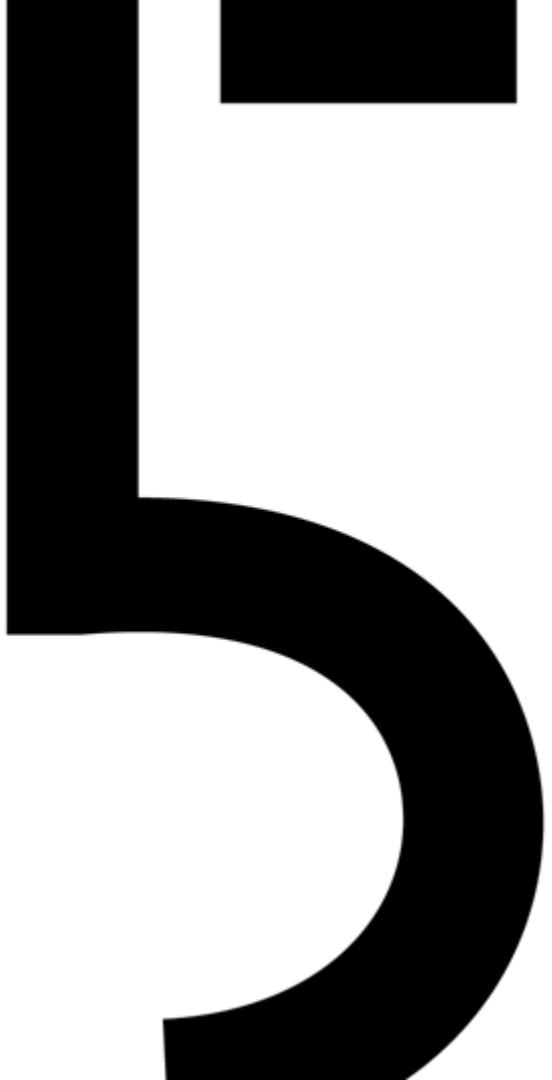
Example

```
#!/bin/bash
backup_file=/tmp/etc.tar.gz

[ $(id -u) -eq 0 ] || { echo "run ${0} as root" >&2 ; exit 1 ; }
[ -z "${1}" ] && { echo "usage: ${0} [now] | [restore]" >&2; exit 1 ; }

case "${1}" in
  now)
    echo "starting backup..."
    tar -czf "${backup_file}" -C /etc . 2>/dev/null
    ;;
  restore)
    echo "restoring backup to /tmp/etc..."
    mkdir -p /tmp/etc
    tar -xzf "${backup_file}" -C /tmp/etc
    ;;
  *)
    # dit is de default
    echo "usage: ${0} [now] | [restore]"
esac
```

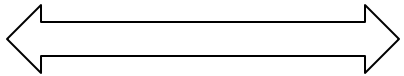
Functions



Functions

- functions can be defined as follows:

```
myFunction() {
  echo ${1}
}
```



```
function myFunction {
  echo ${1}
}
```

2 possibilities

function call

```
myFunction hello
```

Parameters

- no need to declare the parameters
- just use ${1}, ${2}, ..., ${9}
- when calling the function no brackets are used
- a function is in fact like a script within a script

Parameters

```
$ nl param_test.sh
     1  #!/bin/bash
     2  param_test() {
     3      echo "\$1 in function: $1"
     4  }
     5  echo "\$1 in main   : $1"
     6  param_test B
$ ./param_test.sh A
$1 in main   : A
$1 in function: B
```

Parameters

```
$ nl param_test.sh
     1  #!/bin/bash
     2  param_test() {
     3      echo "\$1 in function: $1"
     4  }
     5  echo "\$1 in main   : $1"
     6  par_test B
$ ./param_test.sh A
$1 in main   : A
./param test.sh: line 6: par test: command not found
```

Application in backup script

```
#!/bin/bash
today=$(date +%Y%m%d)
backup_dir=/tmp
backup_file=backup${today}.tgz
source_dir="/root"
logfile=/var/log/backup.log

fatalError() {
  echo "${1}" 1>&2
  [ -w ${logfile} ] && echo "${today}: ${1}" >>${logfile}
  exit 1
}

[ $(id -u) -eq 0 ] || fatalError "Execute as root!"
echo "Backing up to ${backup_file}..."
tar -zcf "${backup_dir}/${backup_file}" "${source_dir}" "${@}" 2>/dev/null \
|| fatalError "Cannot create archive"
sync
echo "${today}: backup successful" >>${logfile}
echo "Done."
```

Debugging – applying best practices?

- Do you want to have your code reviewed?
  Use shellcheck!

  Install:

  ```
  sudo dnf install -y epel-release
  sudo dnf install shellcheck
  ```

  Use: `shellcheck script`

  Use of shellcheck is strongly recommended.

Exercises

Exercise 9.8

9.8 Host monitoring

1. Create a file named **"pinghosts"** that contains the following:

```
www.kdg.be
this.does.not.exist
www.google.be
```

   Write a script named **"pinghosts.sh"** that pings the hosts in this file one by one (once, see `man ping`). The next host is pinged every second, and this continues in an infinite loop. No output should appear on the screen, but write the following to **"/tmp/loghosts"**:

```
Mon Dec 3 07:52:26 CET 2018;www.kdg.be;OK
Mon Dec 3 07:52:27 CET 2018;this.does.not.exist;NOK
```

2. Write a script named **"ping_stats.sh"** that counts how many "OK" and how many "NOK" entries appear in **"/tmp/loghosts"** and outputs the following:

```
Ping statistics:
OK: 186
NOK: 93
```

Karel de Hogeschool

Exercises

- KdG
  - Exercises 9.1 - 9.10
- RedHat
  - none

KdG Karel de Grote Hogeschool