Linux Lab Unit 3
Errorcode, Shellscripting

Christian Kniep

Internation Center of Applied Technologies Bandung

11. August 2010

Table of content

Create sheudle-Environment

- instead of execute the commands you could write them in a textfile and execute them once

Create sheudle-Environment

- instead of execute the commands you could write them in a textfile and execute them once
- create a file `myScript.sh` in your home thats supposed to do
  1. Change to `/var/linuxLab/unit3`

Create sheudle-Environment

- instead of execute the commands you could write them in a textfile and execute them once
- create a file `myScript.sh` in your home thats supposed to do
  1. Change to `/var/linuxLab/unit3`
  2. create a directory thats named with your username

Create sheudle-Environment

- instead of execute the commands you could write them in a textfile and execute them once
- create a file `myScript.sh` in your home thats supposed to do
  1. Change to `/var/linuxLab/unit3`
  2. create a directory thats named with your username
  3. create folders 'monday' to 'friday'

errorcode

- Every command you execute gives back an errorcode from 0-255

errorcode

- Every command you execute gives back an errorcode from 0-255
- if everything went alright it will be 0

errorcode

- Every command you execute gives back an errorcode from 0-255
- if everything went alright it will be 0
- The other values are free to set

errorcode

- Every command you execute gives back an errorcode from 0-255
- if everything went alright it will be 0
- The other values are free to set
- You can get the EC to the variable

```
$ ls -l unit3.tex
-rw-r--r--  1 kniepbert  staff  1245 10 Aug 21:34 unit3.te
$ echo $?
0
$ ls -l unitX.tex
ls: unitX.tex: No such file or directory
$ echo $?
1
```

better way to check

- To test in the filesystem there is a better way...

better way to check

- To test in the filesystem there is a better way...
- You wouldnt get output, so its easier to handle

better way to check

- To test in the filesystem there is a better way...
- You wouldnt get output, so its easier to handle
- say hello to `test`

```
$ test -e unit3.tex
$ echo $?
0
$ test -e unit3.texs
$ echo $?
1
```

better way to check

- To test in the filesystem there is a better way...
- You wouldnt get output, so its easier to handle
- say hello to `test`

```
$ test -e unit3.tex
$ echo $?
0
$ test -e unit3.texs
$ echo $?
1
```

- For all the different test read `man test`

variable=value

- To assign a varbiable with normal values type:

```
$ var=1
$ echo ${var}
1
$ var = 1
-bash: var: command not found
$ var="Hello World"
$ echo ${var}
Hello World
```

var=‘cmd‘

- To assing the **stdout** use `var=‘cmd‘`

var=‘cmd‘

- To assing the **stdout** use `var=‘cmd‘`
- The stderr will not be assing

```
$ ls
unit1.tex unit2.tex unit3.tex
$ var=‘ls‘
$ echo ${var}
unit1.tex unit2.tex unit3.tex
```

basic

- it should look like:

```
if [ CONDITION ]
    then
        CONSEQUENCE
    else
        ALTERNATIVE
    fi
```

example

- if file exists then echo yes, no instead

```
$ touch test.txt
$ if [ -e test.txt ]
>     then
>         echo 'yes'
>     else
>         echo 'no'
>     fi
yes
```

example compare variables

- some variable-comparisons

```
$ x=1
$ y=2
$ if [ x == y ]
>     then
>         echo 'yes'
>     else
>         echo 'no'
>     fi
no
```

example compare variables

- some variable-comparisons

```
$ x=1
$ y=2
$ if [ x == y ]
>     then
>         echo 'yes'
>     else
>         echo 'no'
>     fi
no
```

- all possible conditions in `man test`

example compare variables

- some variable-comparisons

```
$ x=1
$ y=2
$ if [ x == y ]
>     then
>         echo 'yes'
>     else
>         echo 'no'
>     fi
no
```

- all possible conditions in `man test`
- note that the condition 'string-equal' is describted as '=', usualy its '==' which works in bash also.