DEPARTAMENTO
DE SISTEMAS
INFORMÁTICOS

Introduction to shell-scripts

Enrique Arias
Universidad de Castilla–La Mancha

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Main objectives

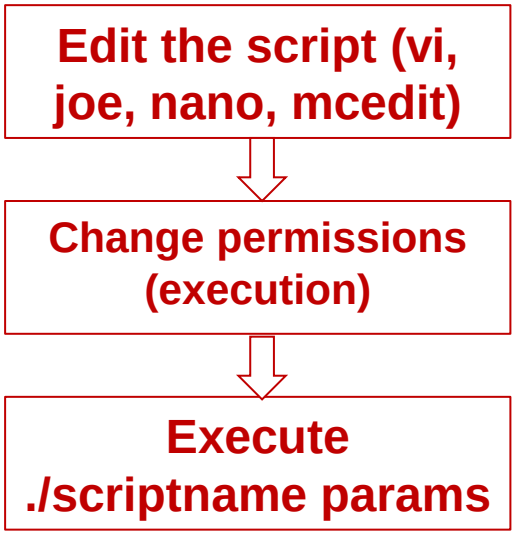
- Programming shell-scripts.
- Flow control operators in shell.

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Shell scripts

- Shell-script is a file containing Shell commands.
- To make more powerful a Shell-script there is an associated language.
- Flowchart of Shell-scripting



Contents

- Main objectives
- Shell scripts
- If-then-else constructs
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

If-then-else construction

- if command
  then command
  [else command]
  fi
- If the evaluation of the command associated to if returns errorlevel 0 then executes then section if not else section.
- Example

```
#!/bin/bash
# Comment
#
if ls -l exists.txt
        # ls -l returns errorlevel=0 if file exists.txt really exists
        # then it will enter in then section
        then echo “That Works ok"
fi
```

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Test command

- Checks file types and compares values.
- test EXPRESSION or [EXPRESSION]
- Expression
  - EXPRESSION1 -a EXPRESSION2 → both EXPRESSION1 and EXPRESSION2 are true
  - EXPRESSION1 -o EXPRESSION2 → either EXPRESSION1 or EXPRESSION2 is true

Test command

- Expression
  - -n STRING → the length of STRING is nonzero
  - -z STRING → the length of STRING is zero
  - STRING1 = STRING2 → the strings are equal
  - STRING1 != STRING2 → the strings are not equal

Test command

- Expression
  - INTEGER1 -eq INTEGER2 INTEGER1 → is equal to INTEGER2
  - INTEGER1 -ge INTEGER2 INTEGER1 → s greater than or equal to INTEGER2
  - INTEGER1 -gt INTEGER2 INTEGER1 → is greater than INTEGER2
  - INTEGER1 -le INTEGER2 INTEGER1 → is less than or equal to INTEGER2
  - INTEGER1 -lt INTEGER2 INTEGER1 → is less than INTEGER2
  - INTEGER1 -ne INTEGER2 INTEGER1 → is not equal to INTEGER2

Test command

- Expression
  - INTEGER1 -eq INTEGER2 → NTEGER1 is equal to INTEGER2
  - INTEGER1 -ge INTEGER2 → INTEGER1 is greater than or equal to INTEGER2
  - INTEGER1 -gt INTEGER2 → INTEGER1 is greater than INTEGER2
  - INTEGER1 -le INTEGER2 → INTEGER1 is less than or equal to INTEGER2
  - INTEGER1 -lt INTEGER2 → INTEGER1 is less than INTEGER2
  - INTEGER1 -ne INTEGER2 → INTEGER1 is not equal to INTEGER2

Test command

- Expression
- -b FILE → FILE exists and is block special
- -c FILE → FILE exists and is character special
- -d FILE → FILE exists and is a directory
- -e FILE → FILE exists
- -f FILE → FILE exists and is a regular file
- -g FILE→ FILE exists and is set-group-ID
- -h FILE → FILE exists and is a symbolic link (same as -L)
- -k FILE → FILE exists and has its sticky bit set
- -p FILE → FILE exists and is a named pipe
- -r FILE → FILE exists and read permission is granted
- -s FILE → FILE exists and has a size greater than zero
- -t FD → file descriptor FD is opened on a terminal
- -u FILE → FILE exists and its set-user-ID bit is set
- -w FILE → FILE exists and write permission is granted
- -x FILE → FILE exists and execute (or search) permission is granted

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Exit

- The exit statement is used to exit from the shell script with a status
- exit [N] being N the status
- Example

```
If ls –l exists.txt
  then
        echo “The file exists”
        exit 0
   else
        echo “The file does not exist”
        exit 1
fi
```

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Shell variables for scripts

$?	Specifies the exit value of the last command executed. 0 indicates successful completion.
$$	Identifies the process number of the current process
$!	Specifies the process number of the last process run in the background using the & terminator.
$#	Specifies the number of positional parameters
$@	Expands the positional parameters, beginning with **$1**
$0	Command name
$i	Positional paramenter i>0

Shell variables for scripts

- Example

```
#!/bin/bash
#
#  This script indicates if a file, entered as parameter, exists or not
#
if [ $# -ne 1 ]
   then
        echo “Incorrect number of parameters"
        exit
fi
if ls -l $1 >/dev/null 2>/dev/null
       then echo “The file exists"
       else echo “The file does not exist"
fi
```

Exercise 1

- Write a script with one parameter to carry out the following functionality
  - Check if the parameter has really passed to the script. If not, print a message.
  - The parameter must be the name of a file. Check if this file is or not in the current directory. Otherwise, print a message.
  - In case of the parameter is a file, check if it is empty or not with a corresponding message.
  - In case of the parameter is a directory, list the directory.

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Iterative constructs: for, while and until

- for
  - the general syntax of the for loop is

for <variable> in <list>
do
    <statement block>
done

  - The <variable> can be any shell variable and will be used by the loop to hold the current value of those given in the <list>
  - <list> could be
    - written explicitly
    - a file name pattern
    - a shell command that generates
    - If no list is indicated, it is considered to be formed by the arguments passed to the shell-script.

Iterative constructs: for, while and until

- for
  - Examples

```
#!/bin/bash
#
for VAL in 1 2 3 4
do
    echo $VAL
done
```

Iterative constructs: for, while and until

- for
  - Examples

```
#!/bin/bash
#
#
num=0;
for i in *
do
    echo $num $i
    num=‘expr $num + 1‘
done
```

Iterative constructs: for, while and until

- for
  - Examples

```
#!/bin/bash
#
#
for i in $@ #(It is the same that "for i")
do
   echo $i
done

# Alternative
num=$#
for ((i=0; i<num; i++ ))
do
   echo $1
   shift
done
```

Iterative constructs: for, while and until

- while
  - The general syntax of the while loop is

while <condition>

do

  <statement block>

Done

  - The <condition> can be any shell condition (-i.e. anything that evaluates to a false (0) or true (non-zero) value).
  - Whilst the <condition> remains true, the <statement block> will be executed - until the <condition> is false (zero).

Iterative constructs: for, while and until

- while
  - Example

```
#!/bin/bash
# Request a string entered by keyboard and print it
while [ -z $string ]
do
   echo “Please, introduce a string"
   read string
done
echo $string
```

Iterative constructs: for, while and until

- until
  - The general syntax of the while loop is

until <condition>

do

  <statement block>

Done

  - The <condition> can be any shell condition (-i.e. anything that evaluates to a false (0) or true (non-zero) value).
  - The <statement block> will be executed whilst the <condition> remains false; when the <condition> becomes true (non-zero), the loop exits.

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Break

- break [number]
- break exits from a for or while loop
- break exits from the given number of enclosing loops.
- break always exits with an exit status of zero.

Break

- Example

```
#!/bin/bash
#
#Print "1 0“ and "1 5". After that, leaves the 2 loops (break 2)
#
for i in 1 2 3
do
    for j in 0 5
    do
       if (test $i -eq 2 -a $j -eq 0)
           then break 2
           else echo "$i $j"
       fi
    done
done
```

Exercise 2

- Write a script that taking as values the files of the HOME directory, organize a structure of directories so that:
  - Files with extension .c are moved to a directory that will be called *prog_C*
  - The files with extension .txt are moved to a directory that will be called *texts*

  The indicated directories will only be created if there are files to be moved to that directory.

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

Case

- The basic syntax of the case statement is to give an expression to evaluate and to execute several different statements based on the value of the expression

```
case word in
  pattern1)
    Statement(s) to be executed if pattern1 matches
    ;;
  pattern2)
    Statement(s) to be executed if pattern2 matches
    ;;
  pattern3)
    Statement(s) to be executed if pattern3 matches
    ;;
  *)
   Default condition to be executed
   ;;
esac
```

Case

- Examples

```
#!/bin/sh

FRUIT="kiwi"

case "$FRUIT" in
  "apple") echo "Apple pie is quite tasty."
  ;;
  "banana") echo "I like banana nut bread."
  ;;
  "kiwi") echo "New Zealand is famous for kiwi."
  ;;
esac
```

Case

- Examples

```
while true
do
    echo “Do you really want to delete the directory $1"
    read answer
    case $answer in
        [yY]*) rmdir $1; break;;
        [nN]*) echo “Don’t delete $1"; break;;
        *) echo “Please, answer yes or not";;
    esac
done
```

Exercise 3

- Write a script called copy that receives a file name as a parameter and then makes a copy of that file to another whose name is the same but followed by an extension that indicates the date on which the copy was made.
- Example: if today is *May 15, 2004* and the file is called *text*, the new file should be called *text.15052004.*
- **Note:** it must be checked that the file passed as a parameter exists and if it does not exist it must be given the right message
- **Help:** check the options of the *date* command.

Contents

- Main objectives
- Shell scripts
- If-then-else construct
- Test command
- Exit
- Shell variables for scripts
- Iterative constructs: for, while and until
- Break
- Case

DEPARTAMENTO
DE SISTEMAS
INFORMÁTICOS

Introduction to shell-scripts

Enrique Arias
Universidad de Castilla–La Mancha