# UNIX

## Lecture 10 : SHELL programming (bash) (Part 4)

Filipe Vasconcelos[1]
[1]ESME, Lille, filipe.vasconcelo@esme.fr

# Outline

1. Pattern Matching
2. String Operations

# Intended Learning Outcomes

**ILO4**

Produce, test, and verify the results of Shell scripts (in Bash) to perform tasks ranging from simple to complex algorithms.

# Outline

1. Pattern Matching
2. String Operations

# Pattern Matching

- In bash, it is possible to perform pattern matching using the **case** keyword.
- The **case** keyword first tests a variable and then attempts to match it successively with each of the patterns.

**Syntax**

```
case WORD in
    pattern_1)           # each pattern ends with )
        <COMMAND1>        # if found, everything that follows is executed...
        <COMMAND2> ;;     # ... up to ;;
    pattern_2)
        <COMMAND1>
        <COMMAND2> ;;
     *)
        <COMMAND1> ;;     # if no test matches
esac                      # ends the block
```

## Example (GitHub): (`exemple_case_cours.sh`)

```
#!/bin/bash
case $1 in
"Monday")
    echo "$1 is just like any $1!";;
"Tuesday")
    echo "$1 is allowed!";;
"Wednesday")
    echo "$1 means it's ravioli day!";;
"Thursday")
    echo "$1 is just Thursday!";;
"Friday")
    echo "$1 means it's almost over!";;
"Saturday")
    echo "$1, it's already Saturday!";;
"Sunday")
    echo "$1 ... nothing!";;
"*")
    echo "Huh?";;
esac
```

# Pattern Matching

It is also possible to use "or" in a ‘case‘ statement for a set of patterns by using the vertical bar to separate each pattern.

**Example (GitHub): (`exemple_case_cours2.sh`)**

```
#!/bin/bash
case $1 in
"Monday"|"Tuesday"|"Wednesday"|"Thursday"|"Friday")
    echo "It's a weekday, sadly :/";;
"Saturday"|"Sunday")
    echo "It's the weekend!!! Yay!";;
esac
```

‘./exemple_case_cours2.sh $(date)‘

‘It’s a weekday, sadly :/‘

# Pattern Matching

Patterns can be constructed using wildcards like * ? [...]

**Example (GitHub): (`exemple_case_cours3.sh`)**

```
case ${X} in
??/??/????)
    echo "I guessed it, it's a date" ;;
[A-Z])
    echo "I guessed it, it's an uppercase letter" ;;
[a-z])
    echo "I guessed it, it's a lowercase letter" ;;
[0-9])
    echo "I guessed it, it's a digit" ;;
esac
```

‘./exemple_case_cours3.sh 16/03/2020‘

‘I guessed it, it’s a date‘

# Pattern Matching (Regular Expression)

It is also possible to test if a string satisfies a regular expression using the ‘if‘ keyword and the ‘= ‘ operator.

**Example (GitHub):** `exemple_romain.sh`

```
if [[ $1 =~ ^(V?(I{0,3})?|I[XV])$ ]] && ![[ $1 =~ ^$ ]]
then
    echo "$1 is a Roman numeral"
else
    echo "$1 is not a Roman numeral"
fi
```

```
./exemple_romain.sh VIIII
```

```
VIIII is not a Roman numeral
```

# Classic Example of Using ‘case‘

<MINTED>

# Outline

1. Pattern Matching
2. String Operations

# Substring Modifier

```
$ string=filipe
```

**Remove Shortest Match from the Left**

```
$ echo ${string#*i}
```

```
lipe
```

**Remove Longest Match from the Left**

```
$ echo ${string##*i}
```

```
pe
```

# Substring Modifier

**Remove Shortest Match from the Right**

```
$ echo ${string%i*}
```

```
fil
```

**Remove Longest Match from the Right**

```
$ echo ${string%%i*}
```

```
f
```

# Substring Extraction

**Extract from Index 3 Onwards**

```
$ echo ${string:3}
```

```
ipe
```

**Extract 2 Characters Starting from Index 3**

```
$ echo ${string:3:2}
```

```
ip
```

# Substring Replacement

### Replace the First Occurrence

```
$ echo ${string/ipe/ou}
```

```
filou
```

### Replace All Occurrences

```
$ echo ${string//i/u}
```

```
fulupe
```

# Substring Deletion

**Delete the First Occurrence**

```
$ echo ${string/e/}
```

```
filip
```

**Delete All Occurrences**

```
$ echo ${string//i/}
```

```
flpe
```

# Delete a caracter from a set

**Deleting the first occurrence of a elements of a set**

```
$ echo ${chaine/[aeiou]/}
```

```
flipe
```

**Deleting all occurrences elements of a set**

```
$ echo ${chaine//[aeiou]/}
```

```
flp
```

# String Transformation

**Lowercase to Uppercase**

```
$ echo ${string^^}
```

```
FILIPE
```

**Uppercase to Lowercase**

```
$ echo ${string,,}
```

# String Generation

```
$ for i in $PWD/{bin,src,lib};do echo $i ;done
```

```
/home/filipe/bin
/home/filipe/src
/home/filipe/lib
```

# basename
## excerpt from the manual

```
NAME
       basename - Remove directory and suffix from filename

SYNOPSIS
       basename NAME [SUFFIX]
       basename OPTION... NAME...

DESCRIPTION
       Mandatory arguments for long options are also mandatory for short options.

       -a, --multiple
              Accept multiple arguments and treat each as a NAME.

       -s, --suffix=SUFFIX
              Remove SUFFIX if present; implies -a.

       -z, --zero
              Terminate each output line with a null character instead of a newline.
```

# Example of using `basename`

```
$ basename /usr/bin/sort
```

```
sort
```

```
$ basename include/stdio.h -s .h
```

```
stdio
```

```
$ basename -a nimp/chaîne1
nimp/chaîne2
```

```
chaîne1
chaîne2
```

```
$ basename https://fr.wikipedia.org
org
```

```
fr.wikipedia.
```

```
$ basename https://fr.wikipedia.org/wiki/Richard_Stallman
```

```
Richard_Stallman
```