P01

Comandos

Comando	Descrição
pwd	
ls (-1 , *xpto , *xpto* , -d , -a , -ll)	
chmod u+x file.txt	Dar permissoes para executar ao ficheiro file.txt
man ls	Acede ao manual do ls
cp	
mv	
mkdir	
rmdir	
history	
cat file.txt	Lista o conteudo do file.txt
tar xfvz file.tar.gz	Extrair o file.tar.gz
tar cfvz xpto.tar.gz	Comprime o ficheiro xpto num tar.gz
who > file.txt	Escreve no file.txt o criador do ficheiro
date >> file.txt	Appends a data ao file.txt
paste dados2.txt dados1.txt > teste.txt	Concatena os 2 ficheiros no teste.txt
ls *da* | wc	Conta o numero de ficheiros com “da” no seu nome
echo $HOME	Imprime o path do diretório

Notas:

- Ter em atenção as diferenças do exercício 11

P02

1)

```
#!/bin/bash
#Primeiro script

msg="Primeira Script. Hell yeah!"
echo $msg
~
```

echo $msg => Imprime o conteudo de msg
echo msg => Imprime “msg”

2)

```
#/!bin/bash
#Watch the spaces!

echo Este       e       um      teste
echo "Este      e       um      teste"
```

No primeiro caso a String é impressa com espaços em vez de TABS

3)

O comando **type** permite saber qual o tipo de um comando

4.b)

```
#!/bin/bash

echo "O meu editor por omissão $BASH"
echo 'O meu editor por omissão $BASH'

echo $(( 5 + 5 ))

(( 5 > 0 )) && echo "cinco é maior do que zero"
today=$(date); echo $today
```

Para executar um **comando** numa string esta tem que estar dentro de aspas.

4.c)

i) **ls**
ii)**ls *a**
iii) **ls -1q -d a*??**
iiii)**ls -1q -d *conf***

5)

```
#!/bin/bash
echo "$#"
echo "Arg 1: $1"
echo "Arg 2: $2"
echo "$*"
echo "$@"
~
~
~
```

O $1 e o $2, sao argumentos passados aquando a invocação da script (**./nome aaa bbb**)

6)

```
#!/bin/bash
# Parameter Expansion

file="$HOME/.bashrc"
echo "File path: $file"
echo "File name: ${file##*/}"
echo "Directory name: ${file%/*}"
~
```

-> shortest match
-> longest match

Ambos apagam a string desde o início

% -> Apaga a string desde o fim

${file##*/} -> Apaga a String desde o ínicio até à última '/'
${file%/*} -> Apaga a String desde o fim até à última '/'

7)

```
#!/bin/bash

echo {1..9}
echo {0,1} {0..9}
```

echo {1..9} -> 1 2 3 4 5 6 7 8 9

8)

```
#!/bin/bash
# Exit status
ping -c 1 www.ua.pt
echo "Exit code: $?" # $? exit code of the last process terminated
ping -c 1 wwwww.ua.pt
echo "Exit code: $?"
```

$? -> dá o exit code do processo corrido anteriormente. Se o processo terminar sem erros o exit code é **0**, caso contrário, tal nao acontece.
File -> Dá o tipo de output de uma script

9)

```
#!/bin/bash

mkdir d && cd d && pwd
echo "----------------"
pwd && rm xpto || echo "I couldn't remove the file"
```

& -> and binário
&& -> and

```
[rd@RD P02]$ ./aula02e09.sh
/home/rd/Área de Trabalho/UA/UA_3_Semestre/SO/P/P02/d
----------------
/home/rd/Área de Trabalho/UA/UA_3_Semestre/SO/P/P02/d
rm: cannot remove 'xpto': No such file or directory
I couldn't remove the file
[rd@RD P02]$
```

Aula 03

```
#!/bin/bash

{
        i=0
        while read line; do
                echo $i : $line
                i=$(($i+1))
        done

}<$1
```

1)

Esta script dá print do ficheiro linha a linha (1 : aaa , 2: bbb)

2)

a)

```
#!/bin/bash


if $1;then
        echo "Verdadeiro"
else
        echo "Falso"
fi
```

Se um comando for built-in ela retorna Verdadeiro (ls-> true , xpto -> 0)

b)

```
#!/bin/bash

if [ $1 = $2 ]; then
        echo "Argumentos Iguais!"
else
        echo "Argumentos Diferentes!"
fi
```

```
#!/bin/bash

if [[ $1 = $2 ]]; then
        echo "Argumentos Iguais!"
else
        echo "Argumentos Diferentes!"
fi
```

A segunda forma é a melhor para comparar strings, por exemplo, uma vez que tem em conta os espaços em branco, enquanto a primeira corta uma frase pelo espaço. Caso queiramos usar apenas [e] temos que escrever:

```
#!/bin/bash

if [ "$1" = "$2" ]; then
        echo "Argumentos Iguais!"
else
        echo "Argumentos Diferentes!"
fi
```

e)

```
#!/bin/bash

if (( $1 > 5 )) && (( $1 < 10 )) ; then
        echo "O número é maior que 5 e menor que 10"
else
        echo " O número NAO é maior que 5 e menor que 10"
fi
```

f)

Para verificar a existência de um ficheiro:

```
#!/bin/bash
#Checking file existence

echo "Checking..."

if [[ -f $1 ]] ; then
        echo "$1 existe!"
else
        echo "$1 nao existe!"
fi
```

Devemos, também, verificar o número de argumentos passados:

```
if [ "$#" -eq 1 ]; then
```

3)

Utilização do case:

```
#!/bin/bash
#This script does a very simple test for checking disk space.
space=$(df -h | awk '{print $5}' | grep % | grep -v Use | sort -n \
        | tail -1 | cut -d "%" -f1 -)

echo "largest occupied space = $space%"

case $space in
        [1-6]*)
                Message="All OK."
        ;;
        [7-8]*)
                Message="Cleaning out. There's a partition that is $space % full."
        ;;
        9[0-8]*)
                Message="Better buy a new disk... One partition is $space % full."
        ;;
        99)
                Message="I'm drowning here! There's a partition at $space %!"
        ;;
        *)
                Message="I seem to be running with an nonexistent disk..."
        ;;

esac
echo $Message
```

Comando	Ação
df -h	Lista os discos do PC, bem como o seu espaço

awk '{print $5}'	Prints argument $5
grep %	Procura todos os dados com "%"
grep -v Use	Apaga a linha "Use"
sort -n	ordenaçao
tail -1	Seleciona o último elemento da lista ordenada
cut -d "%" -f1 -	Apaga o simbolo "%" de todo o texto

c)

```
#!/bin/bash

case $1 in
        [0-99]*)
                echo "O primeiro argumento é valido!"
        ;;

        *)
                echo "O primeiro argumento é inválido!"
                ;;
esac

case "$2" in
         sec*)
                echo "O segundo argumento é valido!"
        ;;
        *)
                echo  "O segundo argumento é inválido!"
                ;;
esac
```

4)

```
#!/bin/bash
# For all the files in a folder, show their properties

if [[ $# < 3 ]] && [[ -d $1 ]] ; then
for file in "$1"/*; do
    if [[ -f "$file" ]]; then
        dirname="${file%/*}/"
        basename="${file:${#dirname}}"

        if [[ $2 == "-r" ]] && [[ ${basename:0:4} == "new_" ]] ;  then
                 mv "$file" "${dirname}${basename:4}"

        elif [[ $2 == "-r" ]] && [[ ${basename:0:4} != "new_" ]] ;  then
                echo "Não se pode remover o prefixo do ficheiro ${basename} !"

        else
                 mv "$file" "${dirname}new_$basename"
        fi

    fi
done

echo""
echo "Nova Lista de Ficheiros: "
ls teste -1
echo ""

else
        echo "Argumentos Inválidos"

fi
```

5)

6)

Diferença entre um while e um until

```
#!/bin/bash
# Wait for a host, given as argument, to come back online.

host=$1

until ping -c 1 "$host" >& /dev/null; do
        echo "$host is still unavailable."
        sleep 5
done;

echo -e "$host is available again.\a"


echo "Vamos tentar com um while..."

while ! ping -c 1 "$host" >& /dev/null; do
        echo "$host is still unavailable."
        sleep 5
done;

echo -e "$host is available again.\a"
```

7)

- **((a = 23))** ---> $a =23
- **((a++))** ---> Post-increment 'a'
- **((a--))** ---> Post-decrement 'a'
- **((++a))** ---> Pre-increment 'a'
- **((--a))** ---> Pre-decrement 'a', C-style.
- **x= $((c +4))** ---> x= c+4

```bash
#!/bin/bash
# Calculate the sum of a series of numbers.

SCORE="0"
SUM="0"
c="0"
while true; do
        echo -n "Enter your score [0-10] ('q' to quit): "
        read SCORE;

        if (("$SCORE" < "0")) || (("$SCORE" > "10")); then
                echo "Try again: "
        elif [[ "$SCORE" == "q" ]]; then
                if [[ $c == 0 ]] ; then
                        echo "Sem Dados."
                        break
                break
        else
                        echo "Sum:$SUM"
                        media=$((SUM / c))
                        echo "Média:$media"
                        break
                fi
        elif [[ "$SCORE" == "r" ]]; then
                echo "Reset"
                SCORE="0"
                SUM="0"
                c="0"
        else
                c=$((c+1))
                SUM=$((SUM + SCORE))
        fi
done

echo "Exiting."
```

8)

Funcionamento de um menu select

```
#!/bin/bash
# select structure to create menus

PS3='Escolhe 1 Opcao: '

select arg in $@; do

        if [[ -n $arg ]]; then
                echo "You picked $arg ($REPLY)."
        else
                echo "A sair..."
                break
        fi


done
```

```
#!/bin/bash
# select structure to create menus

PS3='Escolhe 1 Opcao: '

select arg in "AAA" "BBB" "CCC" ; do

        if [[ -n $arg ]]; then
                echo "You picked $arg ($REPLY)."
        else
                echo "A sair..."
                break
        fi


done
```

Outra forma de passar as hipóteses:

```
PS3='Please enter your choice: '
options=("Option 1" "Option 2" "Option 3" "Quit")
select opt in "${options[@]}"
```