universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Aula Prática Nº 3

Objetivos

Agrupamento de comandos

Condições

Estrutura `if then else fi`

Estrutura de decisão múltipla `case`

Estrutura de repetição `for`

Estruturas de repetição `while` e `until`

Criação de menus com a estrutura `select`

Guião

1. É possível na *bash* agrupar comandos através da utilização dos caracteres `{ … }`. Crie o *script* **`aula03e01.sh`** com o seguinte conteúdo:

```
#!/bin/bash
# Agrupamento de comandos na Bash
{
   i=0
   while read line; do
     echo $i: $line
     i=$(($i+1))
   done
} < $1
```

execute-o passando um ficheiro de texto como argumento e interprete o resultado (consulte http://mywiki.wooledge.org/BashGuide/TestsAndConditionals)

2. Na *bash* é possível tomar decisões com base na utilização da *keyword* `if` em conjunto com a utilização de testes (caracteres `[[ … ]]` e `[ … ]`). A forma mais compacta de utilizar o `if` é a seguinte:

```
if TEST-COMMANDS; then CONSEQUENT-COMMANDS; fi
```

Contudo, dependendo do programador, podemos ter outros estilos:

```
if TEST-COMMANDS
then OTHER-COMMANDS
fi
```

```
if TEST-COMMANDS
then
    OTHER-COMMANDS
fi
```

```
if TEST-COMMANDS; then
    OTHER-COMMANDS
fi
```

a) Crie o *script* **`aula03e02a.sh`** com o seguinte conteúdo, execute-o passando como argumento os comandos *builtin* da *bash* `true` e `false`, o comando `ls` e as constantes `xpto`, `0` e `1` e interprete o resultado:

```
#!/bin/bash
# Conditional block if
if $1 ; then
    echo "Verdadeiro"
else
    echo "Falso"
fi
```

b) Crie o *script* **`aula03e02b.sh`** com o seguinte conteúdo, execute-o passando como argumento duas palavras (*strings*) e interprete o resultado (atenção à utilização do teste):

```
#!/bin/bash
# Conditional block if
if [[ $1 = $2 ]] ; then
    echo "O arg1 é igual ao arg2"
else
    echo "Os args são diferentes"
fi
```

c) Crie o *script* **`aula3e02c.sh`**, idêntico ao anterior, substituindo **`[[`** por **`[`** e **`]]`** por **`]`**[1]. Experimente ambos os *scripts* usando como argumento palavras e também frases (colocando aspas para delimitar cada argumento).

d) Corrija o *script* **`aula03e02c.sh`** para que funcione correctamente quando tem frases como argumentos.

e) Crie o *script* **`aula03e02e.sh`** que escreve a mensagem "numero maior do que 5 e menor do que 10" conforme o valor do argumento.

3. Crie o *script* **`aula03e03.sh`** com o seguinte conteúdo, execute-o passando como argumento o nome de um ficheiro e interprete o resultado:

```
#!/bin/bash
# This script checks the existence of a file
echo "Checking..."
if [[ -f $1 ]] ; then
    echo "$1 existe."
else
    echo "$1 não existe"
fi
echo "...done."
```

a) Altere os *scripts* anteriores de modo a validar o número de argumentos passados, terminando o *script* com uma mensagem se o número de argumentos não for válido.

1 Consultar http://tldp.org/LDP/abs/html/testconstructs.html para verificar as diferenças.

b) Explore a página http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html, nomeadamente a Tabela 7-1 e altere o *script* **`aula03e03.sh`** de modo a obter mais informação sobre o argumento do *script* (por exemplo se é uma diretoria, quais as suas permissões, etc.).

c) Crie o *script* **`aula03e03c.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash
# This script will test if we're in a leap year or not.

if [[ $# = 1 ]]; then
   year=$1
else
   year=$(date +%Y)
fi

if [[ $(($year % 400)) -eq "0" ]]; then
    echo "This is a leap year.  February has 29 days."
elif [[ $(($year % 4)) -eq 0 ]]; then
    if [[ $(($year % 100)) -ne 0 ]]; then
        echo "This is a leap year, February has 29 days."
    else
        echo "This is not a leap year.  February has 28 days."
    fi
else
    echo "This is not a leap year.  February has 28 days."
fi
```

4. No caso de ser necessário realizar múltiplas decisões, a *bash* disponibiliza uma estrutura do tipo `case`. Crie o *script* **`aula03e04.sh`** com o seguinte conteúdo, execute-o e interprete o resultado. Este *script* avalia o espaço em todas as partições do disco do computador e escreve uma mensagem alusiva à partição mais ocupada.

```
#!/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
```

a) Explique com detalhe o funcionamento do comando que preenche a variável `space`.

b) Modifique o *script* de modo que indique também qual a partição com mais espaço em disco.

c) Usando a estrutura **`case`** crie o *script* **`aula03e04c.sh`** que valida os seus argumentos. Considere que o *script* tem 2 argumentos, o primeiro argumento deve ser um número entre 0 e 99 e o segundo argumento deve começar por **`sec`**.

5. É também possível na *bash* realizar tarefas repetitivas. Existem quatro estruturas principais para o efeito:
   - `for NAME in LIST; do COMMANDS; done`
   - `for ((EXPRESSION;EXPRESSION;EXPRESSION)); do COMMANDS; done`
   - `while CONTROL-COMMAND; do COMMANDS; done`
   - `until TEST-COMMAND; do COMMANDS; done`

   a) Crie o *script* **`aula03e05a.sh`** com o seguinte conteúdo, execute-o passando como argumento o caminho para uma pasta e interprete o resultado:

```
#!/bin/bash
# For all the files in a folder, show their properties
for f in $1/*; do
   file "$f"
done
```

   b) Altere o *script* anterior de modo a validar os seus argumentos. Em particular, deve validar o número de argumentos e o tipo de argumento passado, que terá que ser uma diretoria para que o *script* possa ser executado.

   c) Crie um novo *script* que permita mudar o nome a todos os ficheiros de uma pasta, acrescentando-lhe o prefixo **`new_`**. O nome da pasta deve ser passado como argumento. Adicione, depois, a opção **`-r`** ao *script* para remover o prefixo.

6. Crie o *script* **`aula03e06.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

```
#!/bin/bash
#This script opens 4 terminal windows.
i="0"
while [[ $i -lt 4 ]]; do
    xterm &
    i=$(($i+1))
done
```

   a) Crie o *script* **`aula03e06a.sh`** com o seguinte conteúdo, execute-o passando como argumento um endereço IP de um computador e interprete o resultado:

```
#!/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"
```

b) Qual é a diferença entre a estrutura `until` e `while`?

c) Crie 3 versões do *script* **`aula03e06.sh`** de modo a obter o mesmo resultado usando uma estrutura `until` ou cada uma das versões da estrutura `for`.

7. Crie o *script* **`aula03e07.sh`** com o seguinte conteúdo, execute-o e interprete o resultado:

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

SCORE="0"
SUM="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
        echo "Sum: $SUM."
        break
    else
        SUM=$((SUM + SCORE))
    fi
done
echo "Exiting."
```

a) Altere o *script* anterior de modo a apresentar também a média dos valores introduzidos.

b) Adicione a opção da tecla 'r' para reiniciar a contagem e a soma.

8. A Bash tem uma estrutura `select` que é muito útil para criar menus. Crie o *script* **`aula03e08.sh`** com o seguinte conteúdo, execute-o passando um conjunto de argumentos à sua escolha e interprete o resultado:

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

select arg in $@; do
     echo "You picked $arg ($REPLY)."
done
```

a) Interprete cuidadosamente o funcionamento do *script* e explore a redefinição da variável **`PS3`** com vista a alterar a mensagem que aparece ao utilizador.

b) Adicione uma opção para o programa terminar com a escolha de uma opção não válida.