Sistemas Operativos **Ano lectivo 2022/2023**

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 *(consulte http://mywiki.wooledge.org/BashGuide/TestsAndConditionals)*

1. É possível agrupar comandos na *bash* através da utilização dos caracteres `{ … }`. Crie o *script* **`aula03e01.sh`** com o seguinte conteúdo; execute-o passando um ficheiro de texto como argumento e interprete o resultado.

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

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

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

Contudo, pode ser preferida qualquer das formas seguintes, igualmente válidas:

```
if TEST-COMMANDS
then CONSEQ-COMMANDS
fi
```

```
if TEST-COMMANDS
then
    CONSEQ-COMMANDS
fi
```

```
if TEST-COMMANDS; then
    CONSEQ-COMMANDS
Fi
```

a) Crie o *script* seguinte (**`aula03e02a.sh`**) e execute-o passando sucessivamente como argumentos: i) o comando *builtin* `true`; ii) o comando builtin `false`; iii) o comando `ls`; iv) a constante `xpto`; v) a constante `0`; vi) a constante `1`. Interprete os resultados.

```
#!/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**, substituindo no anterior [[..]] por [..]. Usando como argumentos palavras e também frases (colocando aspas para delimitar cada argumento), constate as diferenças de comportamento destas dois formatos de teste (sobre este tópico, sugere-se a consulta de *http://tldp.org/LDP/abs/html/testconstructs.html*).

d) Crie o *script* **aula03e02d.sh**, que deve corrigir o anterior de forma a funcionar correctamente quando recebe frases como argumentos.

e) Crie o script **aula03e02e.sh** que recebe um argumento e, consoante o seu valor, escreve ou não a mensagem "número maior do que 5 e menor do que 10".

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) Aperfeiçoe os *scripts* anteriores, acrescentando, quando aplicáveis, testes de validação do número de argumentos. Sempre que esse número não for válido, a execução deve terminar, e o utilizador deve receber uma mensagem adequada.

b) Altere **aula03e03.sh** de modo a apurar mais dados sobre o argumento (e.g. se é uma directoria e quais as suas permissões). É útil a informação (nomeadamente a Tabela 7-1) disponível em ***http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html***.

c) Crie o *script* **aula03e03c.sh** listado a seguir; execute-o e interprete o resultado:

```
#!/bin/bash
# Testa se ano dado (ou ano actual, se nenhum for dado)
# é bissexto ou comum.
if [[ $# = 1 ]]; then
    year=$1
else
    year=$(date +%Y)
fi
if [[ $(($year % 400)) -eq "0" ]]; then
    echo "Ano bissexto.  Fevereiro tem 29 dias."
elif [[ $(($year % 4)) -eq 0 ]]; then
    if [[ $(($year % 100)) -ne 0 ]]; then
        echo " Ano bissexto.  Fevereiro tem 29 dias."
    else
        echo "Ano comum.  Fevereiro tem 28 dias."
    fi
else
    echo " Ano comum.  Fevereiro tem 28 dias."
fi
```

4. Para facilitar a gestão de múltiplas decisões, a *bash* disponibiliza a estrutura `case`. Ela é usada no *script* seguinte (**`aula03e04.sh`**), que avalia o espaço em todas as partições do disco, para escrever uma mensagem alusiva à partição mais ocupada. Preencha as opções de `case` em falta (assinaladas 'AQUI') de acordo com os critérios em comentário. Execute e interprete o resultado.

```
#!/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
    AQUI )          # espaço < 70%
      Message="All OK."
      ;;
    AQUI )          # 70% <= espaço < 90%
      Message="Cleaning out. One partition is $space % full."
      ;;
    AQUI )          # 90% <= espaço < 99%
      Message="Better buy a new disk.  One partition is $space % full."
      ;;
    AQUI )          # espaço = 99%
      Message="I'm drowning here!  There's a partition at $space %!"
      ;;
    * )
      Message="I seem to be running with a non-existent disk..."
      ;;
esac
echo $Message
```

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

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

   c) Crie o *script* **`aula03e04c.sh`** que, usando **`case`**, valida os seus dois argumentos: o primeiro deve ser um número entre 0 e 99; o segundo deve começar por **`sec`**.

5. É também possível na *bash* realizar tarefas repetitivas. Eis as quatro estruturas principais para esse 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 conteúdo seguinte; 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 começar por validar o número de argumentos (que deve ser 1), e o seu tipo (deve ser o nome de uma directoria).

   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 conteúdo seguinte; 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 conteúdo seguinte; 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) Analise as estruturas `until` e `while` e esclareça bem a diferença entre elas.

   c) Crie três versões do *script* **`aula03e06.sh`** de modo a obter o mesmo resultado usando uma estrutura `until` e 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 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 estrutura `select` é muito útil para criar menus. Crie o *script* **`aula03e08.sh`** com o conteúdo seguinte; 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) Explore a redefinição da variável **`PS3`** com vista a alterar a mensagem ao utilizador.

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