

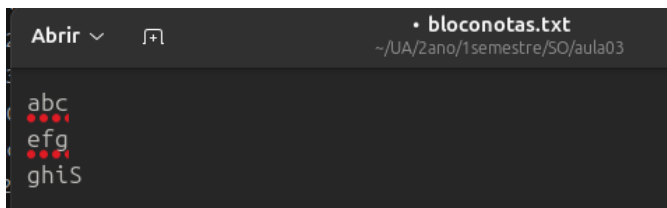
Guião 03 SO

Bernardo Mota Coelho

n.º mec: 125059

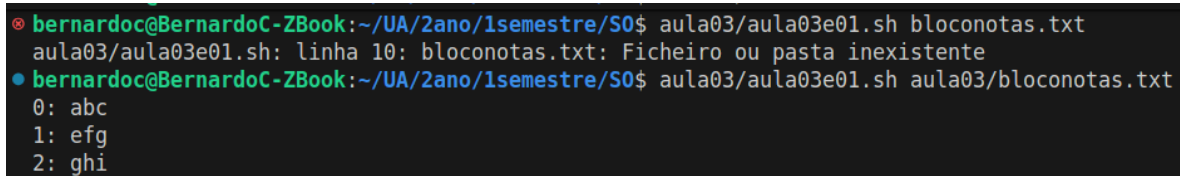
Turma P3

1.



```
Abrir ▾ [F1] • bloconotas.txt
~/UA/2ano/1semestre/SO/aula03

1 abc
2 efg
3 ghi
```

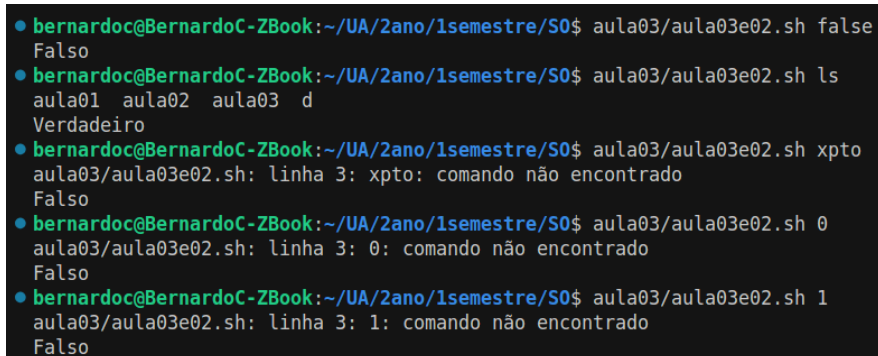


```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e01.sh bloconotas.txt
aula03/aula03e01.sh: linha 10: bloconotas.txt: Ficheiro ou pasta inexistente
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e01.sh aula03/bloconotas.txt
0: abc
1: efg
2: ghi
```

Pelo resultado obtido, é possível verificar que o programa lê o ficheiro .txt linha a linha e imprime, para cada linha, o respetivo número de ordem (começando em 0) seguido do conteúdo da linha.

2.

a)



```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e02.sh false
Falso
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e02.sh ls
aula01 aula02 aula03 d
Verdadeiro
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e02.sh xpto
aula03/aula03e02.sh: linha 3: xpto: comando não encontrado
Falso
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e02.sh 0
aula03/aula03e02.sh: linha 3: 0: comando não encontrado
Falso
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO$ aula03/aula03e02.sh 1
aula03/aula03e02.sh: linha 3: 1: comando não encontrado
Falso
```

O programa verifica se o comando é válido e executável, se for imprime “verdadeiro”, caso contrário imprime “falso”.

b)

```
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02b.sh "abcdef" "ttttt"
Os args são diferentes
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02b.sh "abcdef" "abcdef"
0 arg1 é igual ao arg2
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02b.sh abced
Os args são diferentes
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02b.sh abced abced
0 arg1 é igual ao arg2
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$
```

O script verifica entre 2 inputs se são iguais, se forem iguais imprime “arg1 é igual ao arg2”, se isto não ocorrer então, imprime “Os args são diferentes”.

c)

```
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02c.sh "banana" "banana a"
aula03/aula03e02c.sh: linha 1: : comando não encontrado
aula03/aula03e02c.sh: linha 4: [: demasiados argumentos
Os args são diferentes
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02c.sh "banana" "banana"
aula03/aula03e02c.sh: linha 1: : comando não encontrado
0 arg1 é igual ao arg2
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02c.sh "banana asd asd" "banana as
d asd"
aula03/aula03e02c.sh: linha 1: : comando não encontrado
aula03/aula03e02c.sh: linha 4: [: demasiados argumentos
Os args são diferentes
```

Neste caso, podemos concluir que o uso de “[[]]” é mais robusto do que usar “[]”, pois este último exige a utilização de aspas em variáveis com espaços.

d)

```
1  #!/bin/bash
2  # Conditional block if
3  if [[ $1 = $2 ]]
4  then
5      echo "0 arg1 é igual ao arg2"
6  else
7      echo "Os args são diferentes"
8  fi
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ chmod +x aula03/aula03e02d.sh
● bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02d.sh "banana abd" "banana abd"
0 arg1 é igual ao arg2
```

e)

```
1  #!/bin/bash
2  # Conditional block if
3  if [[ $1 -gt 5 && $1 -lt 10 ]]
4  then
5      echo "número maior que 5 e menor do que 10"
6  else
7      echo "0 número não está entre 5 e 10"
8  fi
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02e.sh 6
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02e.sh 10
0 número não está entre 5 e 10
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02e.sh 11
0 número não está entre 5 e 10
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$ aula03/aula03e02e.sh 9
número maior que 5 e menor do que 10
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0$
```

3.

a)

```
#!/bin/bash
# This script checks the existence of a file

# Validação do número de argumentos

if [[ $# -ne 1 ]]; then # ne = not equal
    echo "Erro: Este script requer exatamente 1 argumento."
    exit 1
fi

echo "Checking..."
if [[ -f $1 ]]
then
    echo "$1 existe."
else
    echo "$1 não existe"
fi
echo "...done."
```

Primeiro, verifica se o argumento é um ficheiro, no caso de verdadeiro imprime " *nome_do_ficheiro* existe, se não for imprime "*nome_do_ficheiro* não existe"

b)

```
1  #!/bin/bash
2  # This script checks the existence of a file and provides detailed information
3
4  # Validação do número de argumentos
5  if [[ $# -ne 1 ]]; then # ne = not equal
6      echo "Erro: Este script requer exatamente 1 argumento."
7      echo "Uso: $0 <caminho>"
8      exit 1
9  fi
10
11  echo "Checking '$1'..."
12
13  # Verificar se existe
14  if [[ -e $1 ]]; then
15      echo "$1 existe."
16
17      # Verificar o tipo de ficheiro
18      if [[ -f $1 ]]; then
19          echo " - É um ficheiro regular"
20      elif [[ -d $1 ]]; then
21          echo " - É uma diretoria"
22      elif [[ -L $1 ]]; then
23          echo " - É um link simbólico"
24      elif [[ -c $1 ]]; then
25          echo " - É um dispositivo de caracteres"
26      elif [[ -b $1 ]]; then
27          echo " - É um dispositivo de blocos"
28      elif [[ -p $1 ]]; then
29          echo " - É um pipe nomeado (FIFO)"
30      elif [[ -S $1 ]]; then
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03b.sh aula03e01.sh
* É possível escrever (writable)
* Executável (executable)
- Tem conteúdo (size > 0)
- És o proprietário
- Pertence ao teu grupo
...done.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$
```

c)

```
1  #!/bin/bash
2  # Testa se ano dado (ou ano actual, se nenhum for dado)
3  # é bissexto ou comum.
4  if [[ $# -eq 1 ]]
5  then
6      year=$1
7  else
8      year=$(date +%Y)
9  fi
10 if [[ $($year % 400) -eq 0 ]]
11 then
12     echo "Ano bissexto. Fevereiro tem 29 dias."
13 elif [[ $($year % 4) -eq 0 ]]
14 then
15     if [[ $($year % 100) -ne 0 ]]
16     then
17         echo " Ano bissexto.  Fevereiro tem 29 dias."
18     else
19         echo "Ano comum.  Fevereiro tem 28 dias."
20     fi
21 else
22     echo " Ano comum.  Fevereiro tem 28 dias."
23 fi
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 2006
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 2007
Ano comum.  Fevereiro tem 28 dias.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 2008
Ano bissexto.  Fevereiro tem 29 dias.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 2025
Ano comum.  Fevereiro tem 28 dias.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 4000
Ano bissexto.  Fevereiro tem 29 dias.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e03c.sh 0
Ano bissexto.  Fevereiro tem 29 dias.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$
```

Basicamente, este script verifica se o ano é bissexto ou comum com if/else.

d)

```
3 > $ aula03e03d.sh
#!/bin/bash
# Testa se ano dado (ou ano actual, se nenhum for dado)
# é bissexto ou comum.

[[ $# -eq 1 ]] && year=$1 || year=$(date +%Y)

[[ $($year % 400) -eq 0 ]] && echo "Ano bissexto. Fevereiro tem 29 dias." || \
[[ $($year % 4) -eq 0 && $($year % 100) -ne 0 ]] && echo "Ano bissexto. Fevereiro tem 29 dias." || \
echo "Ano comum. Fevereiro tem 28 dias."
```

4.

```
aula03 > $ aula03e04.sh
1  #!/bin/bash
2  #This script does a very simple test for checking disk space.
3  space=$(df -h | awk '{print $5}' | grep % | grep -v Use | sort -n \
4  | tail -1 | cut -d "%" -f1 -)
5  echo "largest occupied space = $space%"
6  case $space in
7  [0-6][0-9] ) # espaço < 70%
8      Message="All OK."
9      ;;
10 [7-8][0-9] ) # 70% <= espaço < 90%
11     Message="Cleaning out. One partition is $space % full."
12     ;;
13 9[0-8] ) # 90% <= espaço < 99%
14     Message="Better buy a new disk. One partition is $space % full."
15     ;;
16 99 ) # espaço = 99%
17     Message="I'm drowning here! There's a partition at $space %!"
18     ;;
19 * )
20     Message="I seem to be running with a non-existent disk..."
21     ;;
22 esac
23 echo $Message |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ chmod +x aula03e04.sh
• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e04.sh
./aula03e04.sh: linha 4: erro de sintaxe junto a símbolo "|" inesperado
./aula03e04.sh: linha 4: `| tail -1 | cut -d "%" -f1 -)'
• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e04.sh
largest occupied space = 91%
Better buy a new disk. One partition is 91 % full.
❖ bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$
```

Este script verifica o espaço existente numa partição e dá output de uma mensagem, relativo à quantidade.

a)

df - disk free, mostra o espaço ocupado do disco;

awk - processa o texto por colunas;

grep - filtra as linhas pelo caracter que está a seguir, neste caso “%”, e no caso do “-v Use”, remove as linhas que contenham “Use”;

sort - ordena, neste caso é numérico porque é -n;

tail - mostra o fim, neste caso é só a última linha;

cut - divide a linha em campos.

c)

```
aula03 > $ aula03e04d.sh
1  #!/bin/bash
2  # aula03e04c.sh
3  # Valida 2 argumentos:
4
5  if [[ $# -ne 2 ]]; then
6      echo "Uso: $0 <número 0-99> <string sec...>"
7      exit 1
8  fi
9
10 arg1=$1
11 arg2=$2
12
13 # Validação do primeiro argumento (0-99)
14 case $arg1 in
15     [0-9]|[1-9][0-9]) # 0-9 ou 10-99
16         echo "Primeiro argumento válido: $arg1"
17         ;;
18     *)
19         echo "Erro: o primeiro argumento deve ser um número entre 0 e 99."
20         exit 1
21         ;;
22 esac
23
24 # Validação do segundo argumento (começa por "sec")
25 case $arg2 in
26     sec*) # qualquer coisa que comece por sec
27         echo "Segundo argumento válido: $arg2"
28         ;;
29     *)
30         echo "Erro: o segundo argumento deve começar por 'sec'."
31         exit 1
32         ;;
33 esac
34
35 echo "Ambos os argumentos são válidos!"
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03\$./aula03e04d.sh 99 sec
Primeiro argumento válido: 99
Segundo argumento válido: sec
Ambos os argumentos são válidos!

5.

a)

```
• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e05a.sh /etc
/etc/adduser.conf: ASCII text
/etc/alsa: directory
/etc/alternatives: directory
/etc/anacrontab: ASCII text
/etc/apache2: directory
/etc/apg.conf: ASCII text
/etc/apm: directory
/etc/apparmor: directory
/etc/apparmor.d: directory
/etc/appport: directory
/etc/apt: directory
/etc/avahi: directory
/etc/bash.bashrc: ASCII text
/etc/bash_completion: ASCII text
/etc/bash_completion.d: directory
/etc/bindresvport.blacklist: ASCII text
/etc/binfmt.d: directory
/etc/bluetooth: directory
/etc/brlapi.key: regular file, no read permission
/etc/brltty: directory
/etc/brltty.conf: Unicode text, UTF-8 text, with very long lines (674)
/etc/ca-certificates: directory
```

O programa percorre todos os ficheiros existentes na pasta indicada como argumento na linha de comandos e, para cada um, executa o comando file, que identifica e imprime o tipo de ficheiro.

b)

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

# Validação do número de argumentos
if [[ $# -ne 1 ]]; then
    echo "Erro: Este script requer exatamente 1 argumento."
    echo "Uso: $0 <nome_da_directoria>"
    exit 1
fi

# Validação do tipo de argumento (deve ser uma directoria)
if [[ ! -d $1 ]]; then
    echo "Erro: '$1' não é uma directoria ou não existe."
    echo "Por favor, forneça o nome de uma directoria válida."
    exit 1
fi

for f in $1/*
do
    file "$f"
done
```


c)

```
> $ aula03e03c.sh
#!/bin/bash

remove=false
[[ $1 == "-r" ]] && { remove=true; shift; }
dir=$1

# Verificar argumentos
if [[ -z $dir || ! -d $dir ]]; then
    echo "Uso: $0 [-r] <pasta>"; exit 1
fi

count=0

if $remove; then
    echo "Removendo prefixo 'new_'"
    for f in "$dir"/new_*; do
        [[ -f $f ]] || continue
        newname="${f##*/}"
        newname="${newname#new_}"
        mv "$f" "$dir/$newname" && echo "  ${f##*/} -> $newname" && ((count++))
    done
else
    echo "Adicionando prefixo 'new_'"
    for f in "$dir"/*; do
        [[ -f $f ]] || continue
        base="${f##*/}"
        [[ $base == new_* ]] && continue
        mv "$f" "$dir/new_$base" && echo "  $base -> new_$base" && ((count++))
    done
fi

echo "$count ficheiro(s) processado(s)."
```

6.

a)

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e06a.sh google.com
google.com is available again.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e06a.sh ua.pt
ua.pt is available again.
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ ./aula03e06a.sh ua.ptd
ua.ptd is still unavailable.
```

O programa executa um ping ao host dado como argumento, repetindo a cada 5 segundos, até que este responda.

b)

A estrutura while repete comandos enquanto a condição é verdadeira; a estrutura until repete enquanto a condição é falsa. Assim, until é o oposto lógico de while.

c)

```
#!/bin/bash

#until
i=1
until [ $i -gt 5 ]
do
    echo $i
    i=$((i+1))
done

#for (seq)
for i in $(seq 1 5)
do
    echo $i
done

#for (C)
for (( i=1; i<=5; i++ ))
do
    echo $i
done
```

7.

O programa pede sucessivamente valores entre 0 e 10, acumulando-os numa variável. Se for dado um valor inválido, pede um novo input.

Quando o utilizador introduz 'q', imprime a soma total dos números inseridos e termina.

a)

```
aula03 > $ aula03e07a.sh
1  #!/bin/bash
2  # Calculate the sum of a series of numbers.
3  SCORE="0"
4  SUM="0"
5  COUNTER="0"
6
7  while true
8  do
9      echo -n "Enter your score [0-10] ('q' to quit): "
10     read SCORE;
11     if (($SCORE < "0")) || (($SCORE > "10")); then
12         echo "Try again: "
13     elif [[ "$SCORE" == "q" ]]
14     then
15         echo "Sum: $SUM."
16         echo "Media : $((SUM / COUNTER))"
17         break
18     else
19
20         COUNTER=$((COUNTER+1))
21         SUM=$((SUM + SCORE))
22     fi
23 done
24 echo "Exiting."
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ chmod +x au
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): q
Sum: 15.
Media : 5
Exiting.
```

b)

```
6
7 while true
8 do
9     echo -n "Enter your score [0-10] ('q' to quit) ('r' to reset): "
10    read SCORE;
11    if (("SCORE" < "0") || ("SCORE" > "10")); then
12        echo "Try again: "
13    elif [[ "SCORE" == "q" ]]
14    then
15        echo "Sum: $SUM."
16        echo "Media : $((SUM / COUNTER))"
17        break
18    elif [[ "SCORE" == "r" ]]
19    then
20        echo "Reseting..."
21        SUM=$((0))
22        COUNTER=$((0))
23    else
24
25        COUNTER=$((COUNTER+1))
26        SUM=$((SUM + SCORE))
27    fi
28 done
29 echo "Exiting."
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
• bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$ chmod +x aula03e07b.sh && ./aula03e07b.sh
Enter your score [0-10] ('q' to quit) ('r' to reset): 6
Enter your score [0-10] ('q' to quit) ('r' to reset): 7
Enter your score [0-10] ('q' to quit) ('r' to reset): 8
Enter your score [0-10] ('q' to quit) ('r' to reset): r
Reseting...
Enter your score [0-10] ('q' to quit) ('r' to reset): 1
Enter your score [0-10] ('q' to quit) ('r' to reset): 1
Enter your score [0-10] ('q' to quit) ('r' to reset): 1
Enter your score [0-10] ('q' to quit) ('r' to reset): q
Sum: 3.
Media : 1
Exiting.
○ bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$
```

8.

O script transforma os argumentos recebidos em um menu interativo. O utilizador escolhe um número, e o programa mostra qual a opção escolhida (arg) e o número introduzido (REPLY), pela ordem que foi colocado no terminal.

a)

```
1 #!/bin/bash
2 # select structure to create menus
3
4 PS3="Choose the number you want: "
5 select arg in $@
6 do
7     echo "You picked $arg ($REPLY)."
```

b)

```
aula03 > $ aula03e08.sh
1  #!/bin/bash
2  # select structure to create menus
3
4  PS3="Choose the number you want: "
5  select arg in $@
6  do
7      if [[ -z "$arg" ]];
8      then
9          echo "Invalid Option"
10         break
11     else
12         echo "You picked $arg ($REPLY)."

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS



```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula03$./aula03e08.sh a b c d e
2) b
3) c
4) d
5) e
Choose the number you want: 4
You picked d (4).
Choose the number you want: 6
Invalid Option
```


```