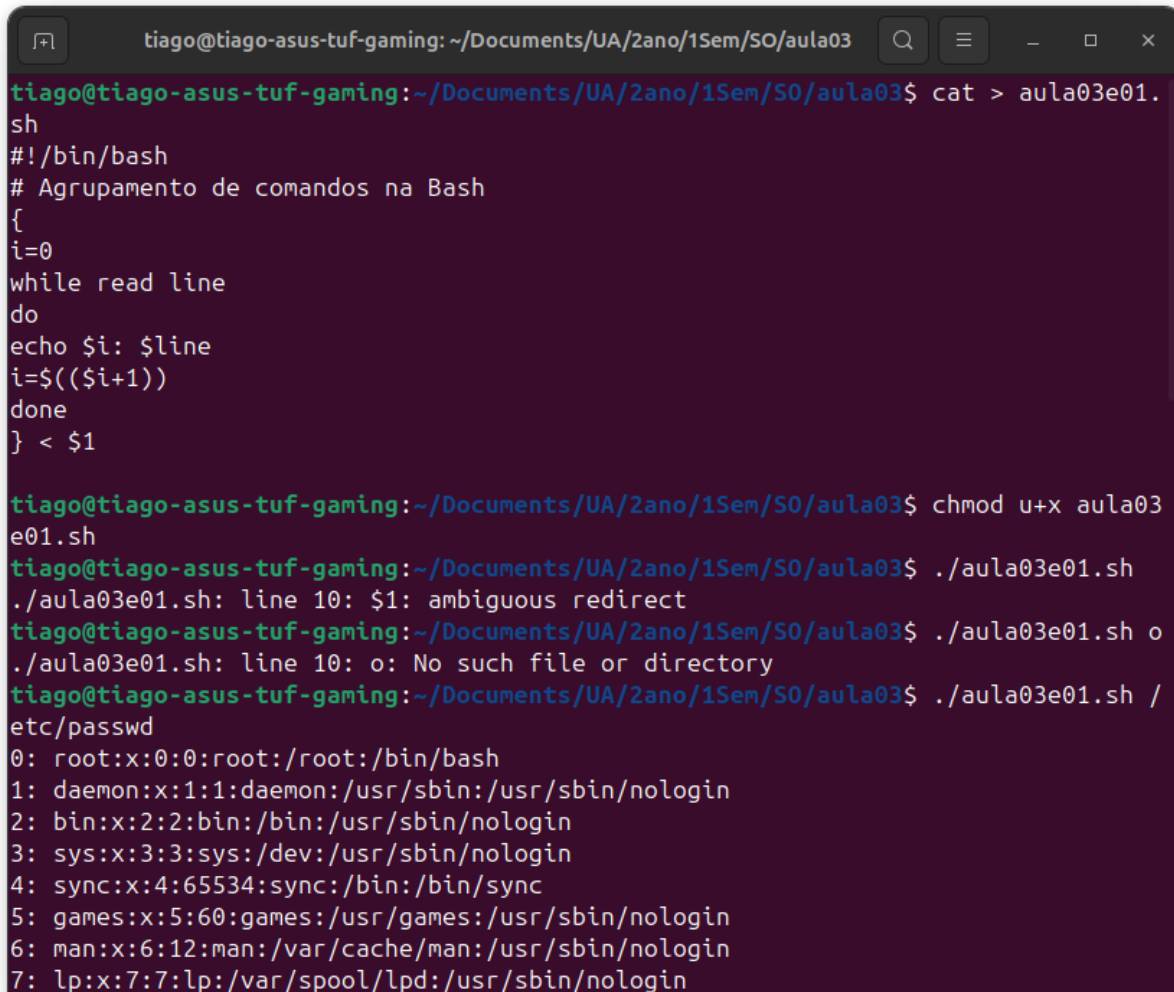


1.



```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ cat > aula03e01.sh
#!/bin/bash
# Agrupamento de comandos na Bash
{
i=0
while read line
do
echo $i: $line
i=$((i+1))
done
} < $1

tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ chmod u+x aula03e01.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e01.sh
./aula03e01.sh: line 10: $1: ambiguous redirect
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e01.sh o
./aula03e01.sh: line 10: o: No such file or directory
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e01.sh /etc/passwd
0: root:x:0:0:root:/root:/bin/bash
1: daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
2: bin:x:2:2:bin:/bin:/usr/sbin/nologin
3: sys:x:3:3:sys:/dev:/usr/sbin/nologin
4: sync:x:4:65534:sync:/bin:/bin/sync
5: games:x:5:60:games:/usr/games:/usr/sbin/nologin
6: man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
7: lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

{ ... } < \$1, tem o mesmo sentido que, por exemplo "cat < passwd", ou seja, vai abrir o ficheiro no primeiro argumento do comando, para o bloco { ... } ser alimento.

While read line do echo \$i: \$line, "i" e "line" são variáveis (i é inteiro e é incrementado), logo por cada linha que houver no código vai fazer (por ex.): 0: linha0; 1: linha1;...

Quando não houver mais linhas no ficheiro: done.

2.

a)

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ cat aula03e02a.sh
#!/bin/bash
# Conditional block if
if "$1"
then
echo "Verdadeiro"
else
echo "Falso"
fi
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02
bash: ./aula03e02: No such file or directory
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
./aula03e02a.sh: line 3: : command not found
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
true
Verdadeiro
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
false
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
ls
aula03e01.sh  aula03e02a.sh  SO-2526_aula03.pdf
Verdadeiro
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
xpto
./aula03e02a.sh: line 3: xpto: command not found
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
0
./aula03e02a.sh: line 3: 0: command not found
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
1
./aula03e02a.sh: line 3: 1: command not found
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02a.sh
1
./aula03e02a.sh: line 3: 1: command not found
Falso
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$
```

Se o argumento for verdadeiro (p/ex. O *builtin* true) o programa entra no bloco CONSEQ-COMMANDS, isto acontece se o argumento for, também, um comando da *shell* que exista. (Os argumentos são interpretados como comandos). Vai dizer “Verdadeiro” a: ls e true; vai dizer “Falso” a: xpto, 0, 1 e ao *builtin* false.

b)

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ cat > aula03e02b.sh
#!/bin/bash
# Conditional block if
if [[ $1 = $2 ]]
then
echo "0 arg1 é igual ao arg2"
else
echo "Os args são diferentes"
fi
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ chmod u+x aula03e02b.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02b.sh
1 1
0 arg1 é igual ao arg2
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02b.sh
1 2
Os args são diferentes
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02b.sh
/etc/passwd/ /etc/passwd
Os args são diferentes
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02b.sh
/etc/passwd/ /etc/passwd/
0 arg1 é igual ao arg2
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$
```

Se os argumentos forem iguais entra no CONSEQ-COMMANDS.

c)

A diferença para parentesis retos duplos [[]] para singulares []:

```
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02c.sh "ola adeus" "ola adeus"
./aula03e02c.sh: line 3: [: too many arguments
Os args são diferentes
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e02b.sh "ola adeus" "ola adeus"
0 arg1 é igual ao arg2
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$
```

d)

```

tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ cat > aula03e02d
#!/bin/bash
# Conditional block if
if [ "$1" = "$2" ]
then
echo "0 arg1 é igual ao arg2"
else
echo "Os args são diferentes"
fi
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ ./aula03e02d.sh
bash: ./aula03e02d.sh: Permission denied
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ chmod u+x aula03e02d.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ ./aula03e02d.sh "ola adeus"
"ola adeus"
0 arg1 é igual ao arg2
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$

```

e)



```

Open  ~  aula03e02e.sh
~/Documents/UA/2ano/1Sem/S0/aula03
aula03e02c.sh  aula03e02d.sh  aula03e02e.sh
#!/bin/bash

if [[ $1 -gt 5 && $1 -lt 10 ]]
then
    echo "número maior do que 5 e menor do que 10"
else
    echo "número menor do que 5 ou maior do que 10"
fi

```

-gt = ">"; -lt = "<"

3.

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ cat > aula03e03.sh
#!/bin/bash
# This script checks the existence of a file
echo "Checking..."
if [[ -f $1 ]]
then
echo "$1 existe."
else
echo "$1 não existe"
fi
echo "...done."
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ chmod u+x aula03e03.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e03.sh
Checking...
não existe
...done.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e03.sh /etc/passwd
Checking...
/etc/passwd existe.
...done.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$
```

b)

```
Open aula03e03.sh
~/Documents/UA/2ano/1Sem/SO/aula03

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

if [[ $# == 1 ]]

then

    echo "Checking..."
    if [[ -f $1 ]]

    then
        echo "$1 existe."
    else
        echo "$1 não existe"
    fi
    echo "...done."

else
    echo "Demasiados argumentos"
fi
```

Apenas verifica um ficheiro por vez.

c) Primeiramente verifica quantos argumentos foram usados como entrada. Se foram usados mais do que 1 argumento, o programa usa o ano da variável \$date (data atual do computador).

-eq = equals; -ne = not equal; sinal de divisão inteira: %.

d) O mesmo programa, só com 1 linha de controlo:



```
Open ▾ [🔍] aula03e03d.sh ~/Documents/UA/2ano/1Sem/SO/aula03
aula03e03c.sh aula03e03d.sh x
# Testa se ano dado (ou ano actual, se nenhum for dado)
# é bissexto ou comum.
if [[ $# -eq 1 ]]
then
    year=$1
else
    year=$(date +%Y)
fi

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

4.

Este programa verifica o espaço existente numa partição, e escreve uma mensagem dependendo da quantidade de espaço ocupado nessa partição

a)

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

awk - processa o texto por colunas;

grep - filtra as linhas pelo caracter processor (%), e quando aparece "-v Use", remove as linhas que contenham "Use";

sort - ordena, neste caso, numericamente, porque aparece -n

tail - mostra o fim (a última linha)

cut - divide a linha em campos

b)

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
line=$(df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{print $5 " " $1}' | sort -n | tail -1)

# Extrair percentagem e partição
space=$(echo $line | awk '{print $1}' | tr -d '%')
partition=$(echo $line | awk '{print $2}')

echo "Partition with highest usage: $partition"
echo "Largest occupied space = $space%"

# Analisar o espaço
case $space in
  [0-6][0-9]) # espaço < 70%
    Message="All OK."
    ;;
  7[0-9]) # 70% <= espaço < 80%
    Message="Cleaning out. One partition is $space% full."
    ;;
  8[0-9]) # 80% <= espaço < 90%
    Message="Better buy a new disk. One partition is $space% full."
    ;;
  9[0-8]) # 90% <= espaço < 99%
    Message="Warning! One partition is $space% full."
    ;;
  99) # espaço = 99%
    Message="I'm drowning here! There's a partition at $space%!"
    ;;
  100)
    Message="Disk totally full!"
    ;;
  *)
    Message="I seem to be running with a non-existent disk..."
    ;;
esac

echo "$Message (Partition: $partition)"

tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03$ chmod u+x aula03e04b.sh
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e04b.sh
Partition with highest usage: /dev/nvme1n1p1
Largest occupied space = 33%
All OK. (Partition: /dev/nvme1n1p1)
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03$
```

c)

```
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03
if [[ $# -ne 2 ]]; then
    echo "Uso: $0 <número 0-99 <string sec...>"
    exit 1
fi

arg1=$1
arg2=$2

#Validação do primeiro argumento (0-99)
case $arg1 in
    [0-9]|[1-9][0-9]) # 0-9 ou 10-99
        echo "Primeiro argumento válido: $arg1"
        ;;
    *)
        echo "Erro: o primeiro argumento deve ser um número entre 0 e 99."
        exit 1
        ;;
esac

#Validação do segundo argumento (começa por "sec")
case $arg2 in
    sec*)
        echo "Segundo argumento válido: $arg2"
        ;;
    *)
        echo "Erro: o segundo argumento deve começar por 'sec'."
        exit 1
        ;;
esac

echo "Ambos os argumentos são válidos"

tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$ chmod u+x aula03e04c.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$ ./aula03e04c.sh
Uso: ./aula03e04c.sh <número 0-99 <string sec...>
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$ ./aula03e04c.sh 50 seca
Primeiro argumento válido: 50
Segundo argumento válido: seca
Ambos os argumentos são válidos
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$ ./aula03e04c.sh 101 zeca
Erro: o primeiro argumento deve ser um número entre 0 e 99.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$ ./aula03e04c.sh 10 zeca
Primeiro argumento válido: 10
Erro: o segundo argumento deve começar por 'sec'.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1sem/50/aula03$
```

d)

```
Open  aula03e04d.sh  aula03e04d.sh
~/Documents/UA/2ano/1sem/50/aula03

#!/bin/bash
# aula03e04d.sh
# This script does a very simple test for checking disk space (using if/elif/else)

# Obter percentagem de ocupação máxima
space=$(df -h | awk '{print $5}' | grep % | grep -v Use | sort -n \
| tail -1 | cut -d "%" -f1 -)

echo "Largest occupied space = $space%"

# Verificar condições
if [[ $space -lt 70 ]]; then
    Message="All OK."
elif [[ $space -lt 90 ]]; then
    Message="Cleaning out. One partition is $space% full."
elif [[ $space -lt 99 ]]; then
    Message="Better buy a new disk. One partition is $space% full."
elif [[ $space -eq 99 ]]; then
    Message="I'm drowning here! There's a partition at $space%!"
else
    Message="I seem to be running with a non-existent disk..."
fi

echo "$Message"
```

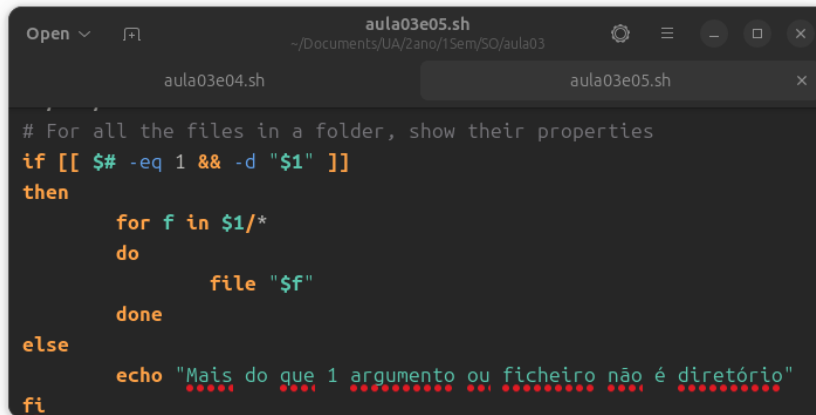
É mais fácil escrever condições numéricas de forma mais direta a usar a estrutura if/else/fi. Porém a estrutura do case é muito mais legível e deixa o script mais simples.

Como neste exercício estamos a trabalhar com valores e intervalos numéricos, acho mais direto e apropriado usar a estrutura do if/else/fi porque torna o código mais flexível.

5.

a) Por cada ficheiro no diretório do argumento, o programa vai fazer *file* desse ficheiro.

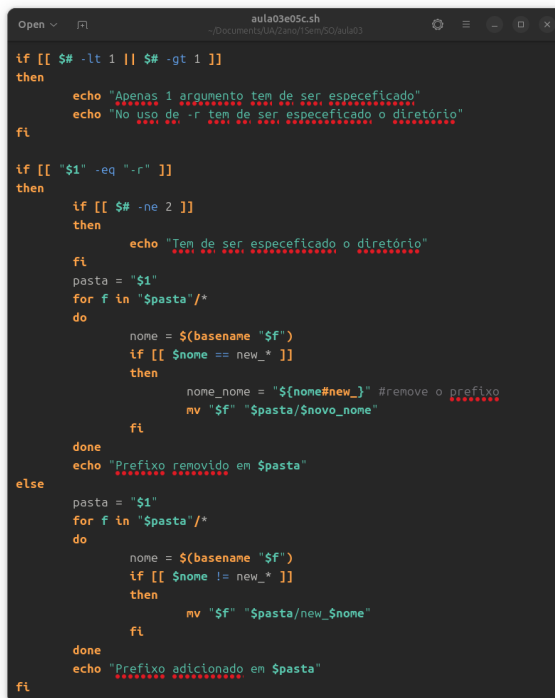
b)



```
Open ▾ [🔍] aula03e05.sh ~/Documents/UA/2ano/1Sem/SO/aula03
aula03e04.sh aula03e05.sh x

# For all the files in a folder, show their properties
if [[ $# -eq 1 && -d "$1" ]]
then
    for f in $1/*
    do
        file "$f"
    done
else
    echo "Mais do que 1 argumento ou ficheiro não é diretório"
fi
```

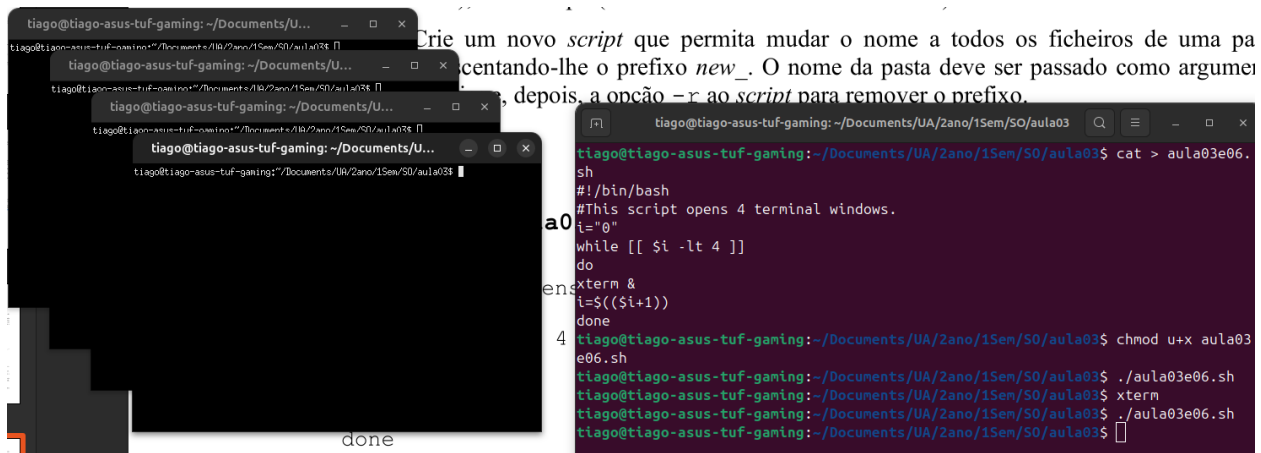
c)



```
Open ▾ [🔍] aula03e05c.sh ~/Documents/UA/2ano/1Sem/SO/aula03
if [[ $# -lt 1 || $# -gt 1 ]]
then
    echo "Apenas 1 argumento tem de ser especeficado"
    echo "No uso de -r tem de ser especeficado o diretório"
fi

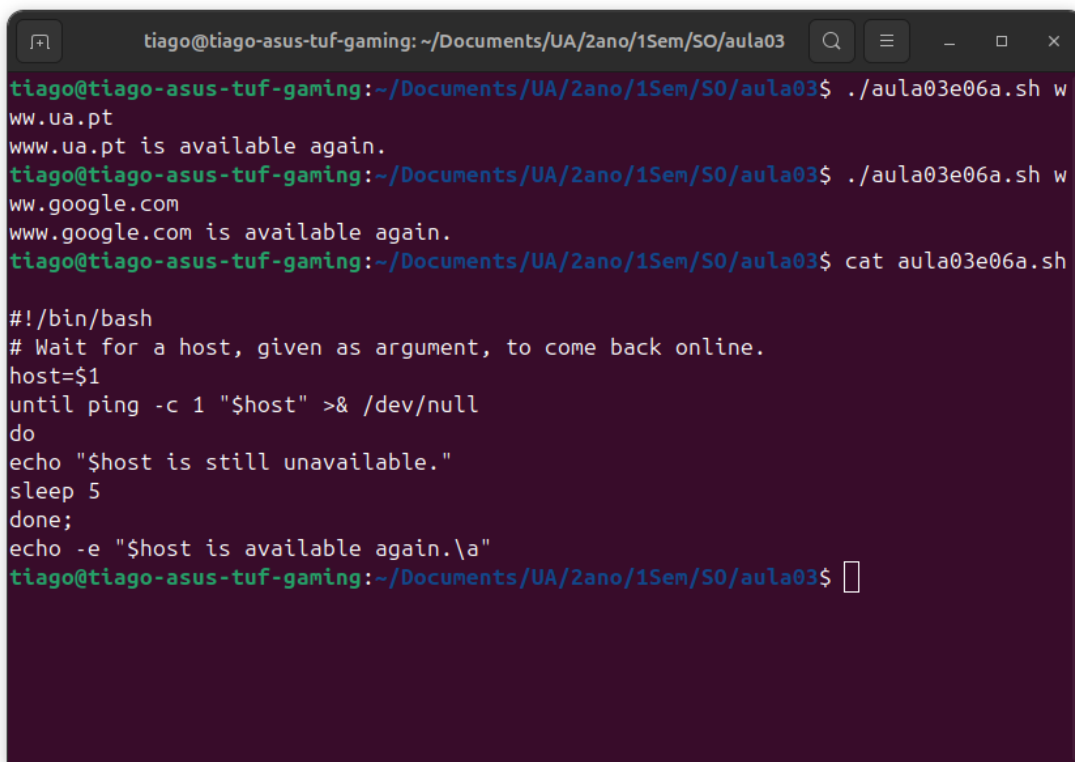
if [[ "$1" -eq "-r" ]]
then
    if [[ $# -ne 2 ]]
    then
        echo "Tem de ser especeficado o diretório"
    fi
    pasta = "$1"
    for f in "$pasta"/*
    do
        nome = $(basename "$f")
        if [[ $nome == new_* ]]
        then
            nome_novo = "${nome#new_}" #remove o prefixo
            mv "$f" "$pasta/$nome_novo"
        fi
    done
    echo "Prefixo removido em $pasta"
else
    pasta = "$1"
    for f in "$pasta"/*
    do
        nome = $(basename "$f")
        if [[ $nome != new_* ]]
        then
            mv "$f" "$pasta/new_$nome"
        fi
    done
    echo "Prefixo adicionado em $pasta"
fi
```

6.



O comando *xterm*, abre um terminal externo. Sabendo isso, o programa entra num loop de quatro iterações e cada iteração abre um terminal novo.

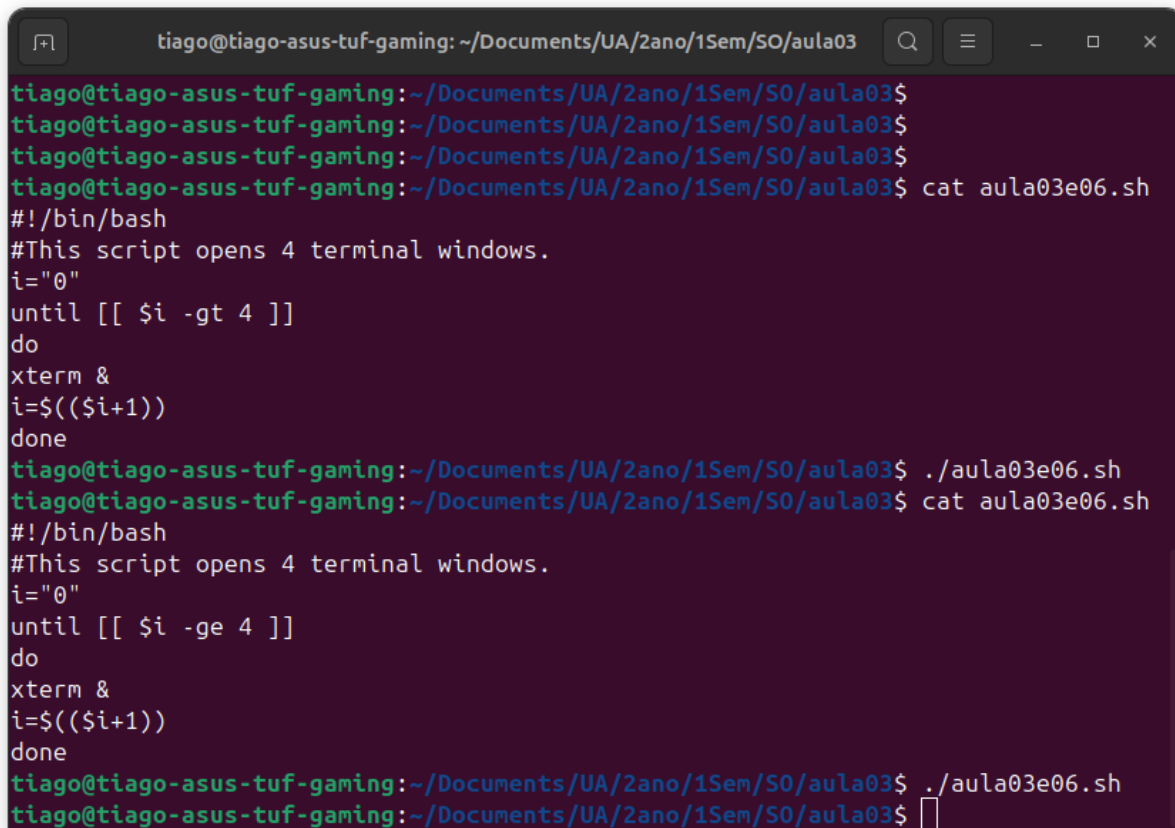
a)



O programa tenta dar *ping* a um endereço, e até que o endereço ser “pingado”, o programa vai escrever “*endereço* (o argumento) is still unavailable” e fica 5 segundos a “dormir”.

b) O until é o inverso lógico do while, isto significa que no while o bloco só é percorrido se a condição for verdadeira enquanto no until o bloco só é percorrido se a condição for falsa.

c)



```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/S0/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ cat aula03e06.sh
#!/bin/bash
#This script opens 4 terminal windows.
i="0"
until [[ $i -gt 4 ]]
do
xterm &
i=$((i+1))
done
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ ./aula03e06.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ cat aula03e06.sh
#!/bin/bash
#This script opens 4 terminal windows.
i="0"
until [[ $i -ge 4 ]]
do
xterm &
i=$((i+1))
done
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$ ./aula03e06.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/S0/aula03$
```

Para funcionar temos de usar o operador lógico oposto ao less than (-lt) que é o greater or equal then (-ge)

7. O programa faz um loop (do while) e enquanto a entrada não for "q" ele vai somando os valores inbutidos, e no final faz echo do resultado.

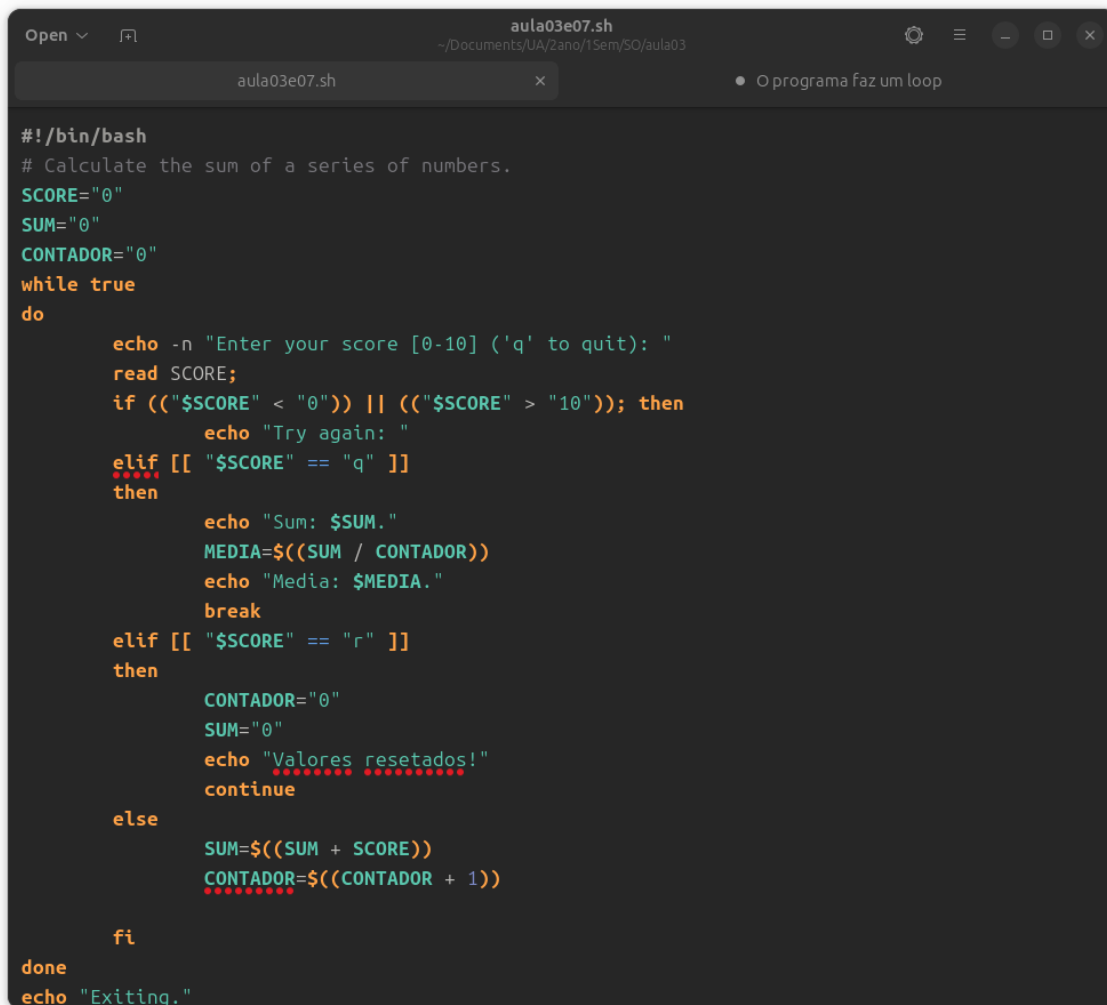
a)

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ chmod u+x aula03e07.sh
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e07.sh
Enter your score [0-10] ('q' to quit): 4
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): 4
Enter your score [0-10] ('q' to quit): q
Sum: 13.
./aula03e07.sh: line 16: echi: command not found
Exiting.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e07.sh
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): 7
Enter your score [0-10] ('q' to quit): 7
Enter your score [0-10] ('q' to quit): 5
Enter your score [0-10] ('q' to quit): q
Sum: 24.
Media: 6
Exiting.
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$
```

```
Open  aula03e07.sh
~/Documents/UA/2ano/1Sem/SO/aula03

#!/bin/bash
# Calculate the sum of a series of numbers.
SCORE="0"
SUM="0"
CONTADOR="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: $SUM."
        MEDIA=$((SUM / CONTADOR))
        echo "Media: $MEDIA"
        break
    else
        SUM=$((SUM + SCORE))
        CONTADOR=$((CONTADOR + 1))
    fi
done
echo "Exiting."
```

b)



```
#!/bin/bash
# Calculate the sum of a series of numbers.
SCORE="0"
SUM="0"
CONTADOR="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: $SUM."
        MEDIA=$((SUM / CONTADOR))
        echo "Media: $MEDIA."
        break
    elif [[ "$SCORE" == "r" ]]
    then
        CONTADOR="0"
        SUM="0"
        echo "Valores resetados!"
        continue
    else
        SUM=$((SUM + SCORE))
        CONTADOR=$((CONTADOR + 1))
    fi
done
echo "Exiting."
```

Fazemos uma nova condição se o valor do \$SCORE for "r".

8.

O programa faz "display" de todos os argumentos que colocamos na execução, na qual podemos escolher 1 desses argumentos e faz echo do argumento escolhido \$arg e o número correspondente a esse argumento \$REPLY

```
tiago@tiago-asus-tuf-gaming: ~/Documents/UA/2ano/1Sem/SO/aula03
tiago@tiago-asus-tuf-gaming:~/Documents/UA/2ano/1Sem/SO/aula03$ ./aula03e08.sh a
b c d
1) a
2) b
3) c
4) d
#? 1
You picked a (1).
#? 2
You picked b (2).
#? 3
You picked c (3).
#? 4
You picked d (4).
#? 5
You picked (5).
#? a
You picked (a).
#? b
You picked (b).
#? 
```

b)

```
Open  aula03e08.sh
~/Documents/UA/2ano/1Sem/SO/aula03
O programa faz display

#!/bin/bash
# select structure to create menus
select arg in $@
do
    if [[ -z "$arg" ]]; then
        echo "Opção inválida. A sair..."
        break
    fi
    echo "You picked $arg ($REPLY)."
done
```

Nota: “-z” para verificar se está vazio