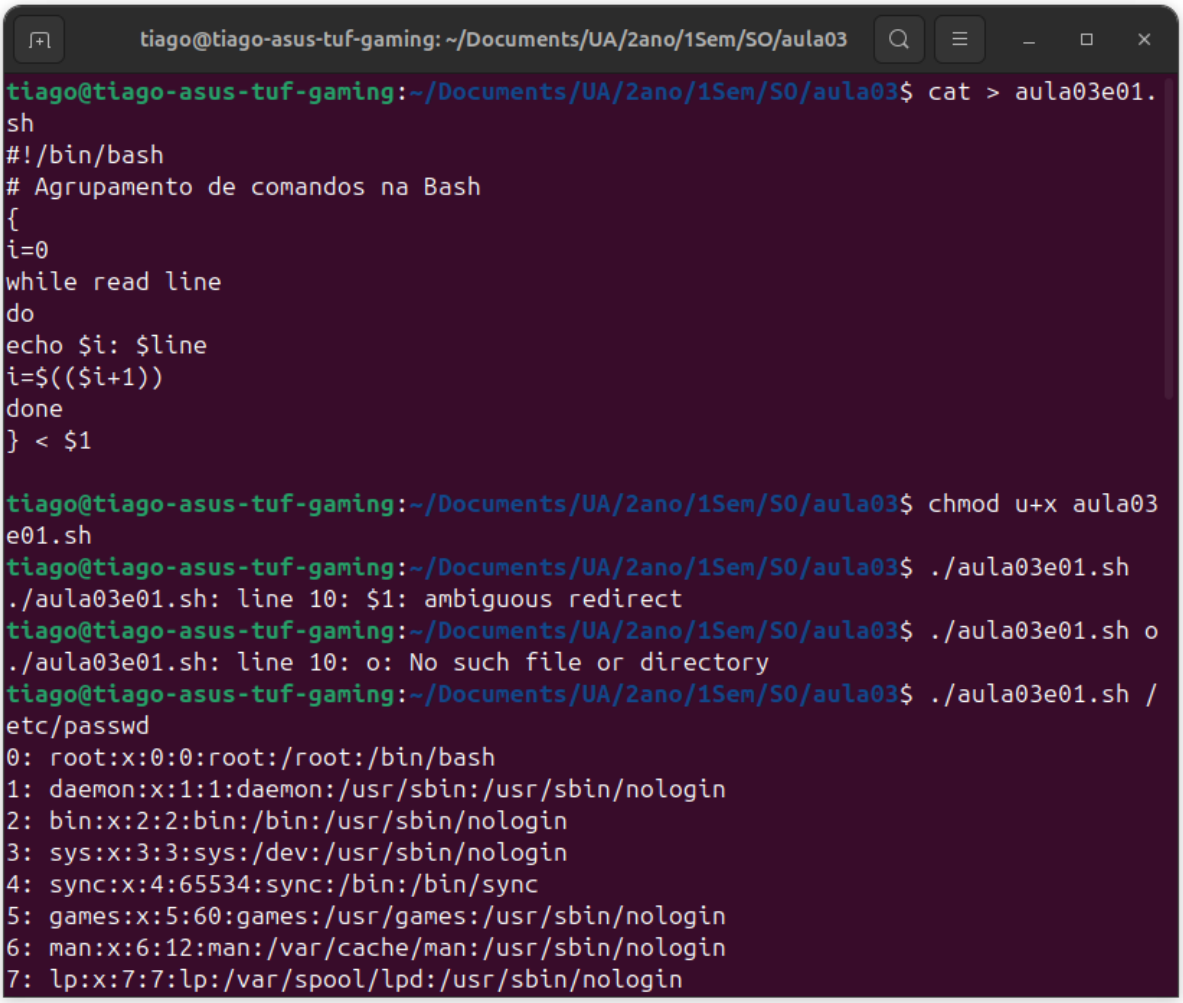
1.



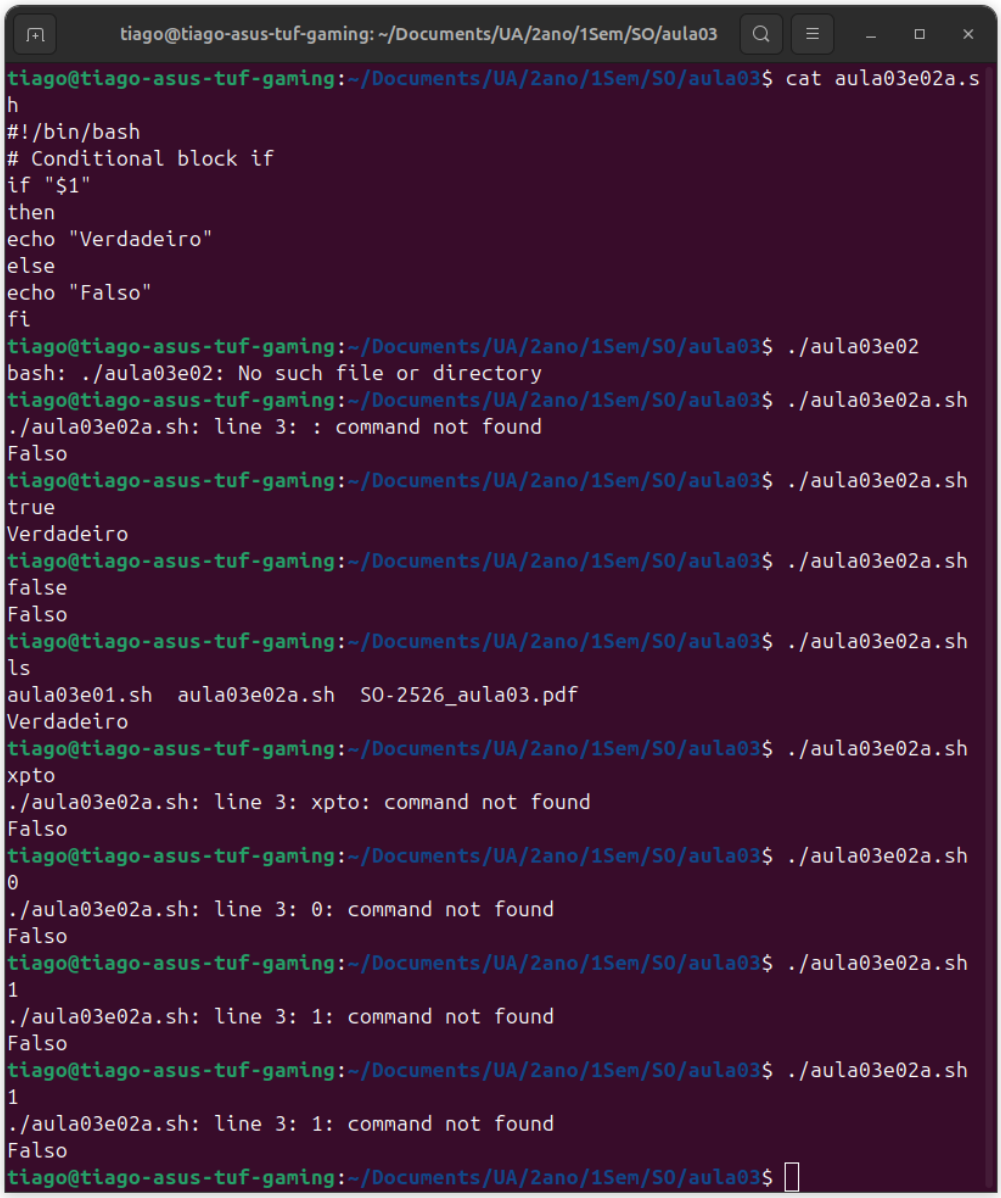
{ ... } < $1, tem o meso sentido que, por exemplo ”cat < passwd”, ou seja, vai abrir o ficheiro no primeiro argumento do comando, para o bloco { ... } ser alimento.

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.

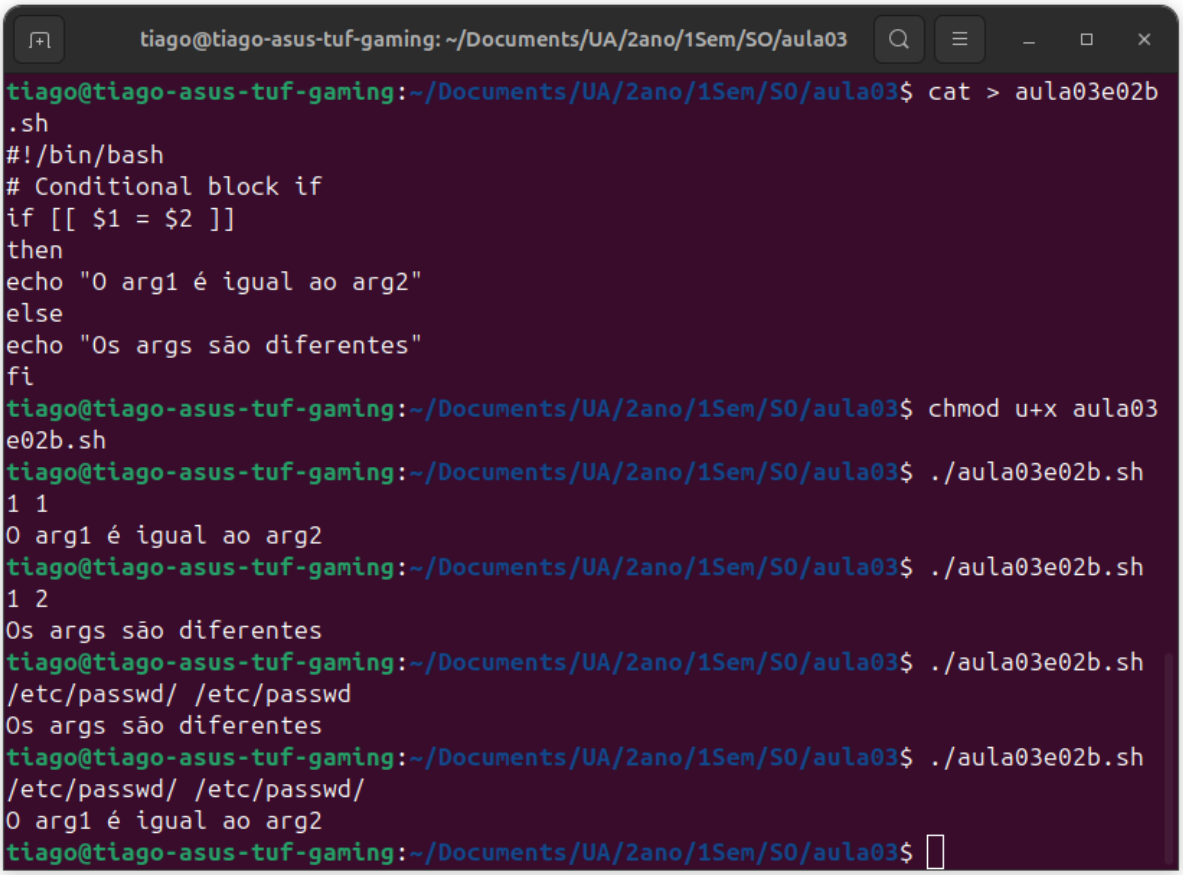
2.

a)



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 intrepertados como comandos). Vai dizer “Verdadeiro” a: ls e true; vai dizer “Falso” a: xpto, 0, 1 e ao *builtin* false.

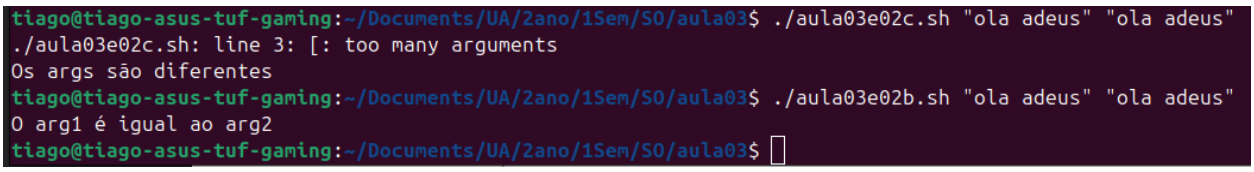
b)



Se os argumentos forem iguais entra no CONSEQ-COMMANDS.

c)

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



d)

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

e)

-gt = “>”; -lt = “<”

3.

b)

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:

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;

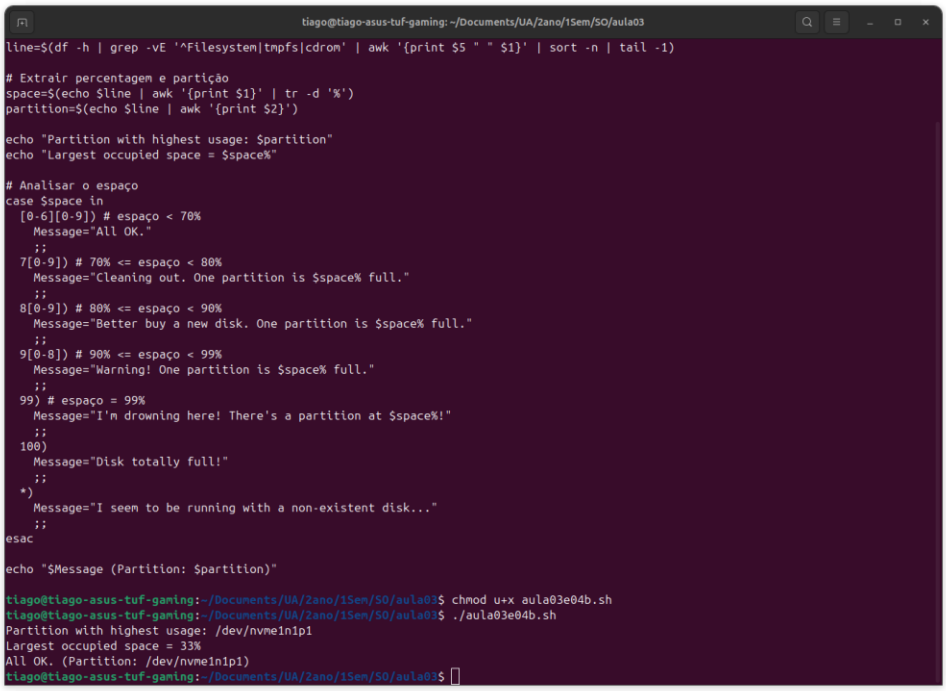
grep - filtra as linhas pelo caracter processor (%), e quando aparece "-v Use", remove as linhas que contenham "Use";

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

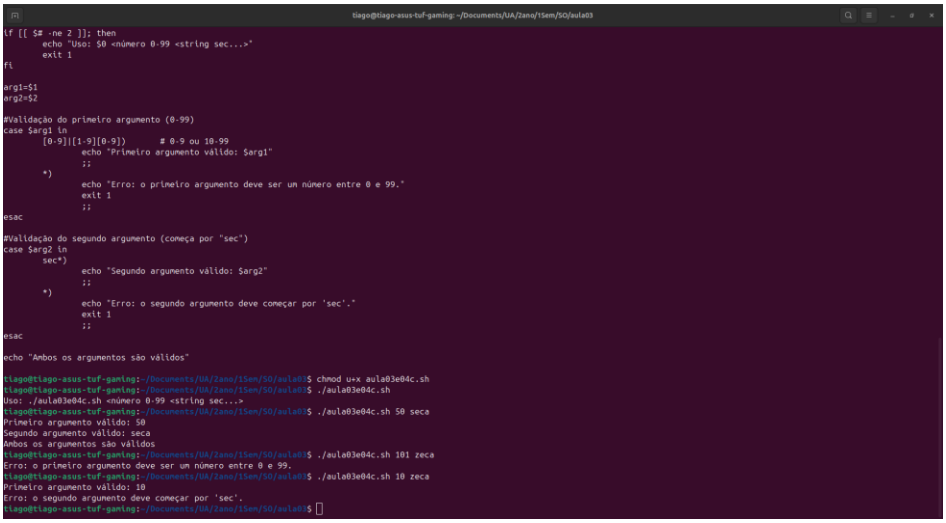
tail - mostra o fim (a última linha)

cut - divide a linha em campos

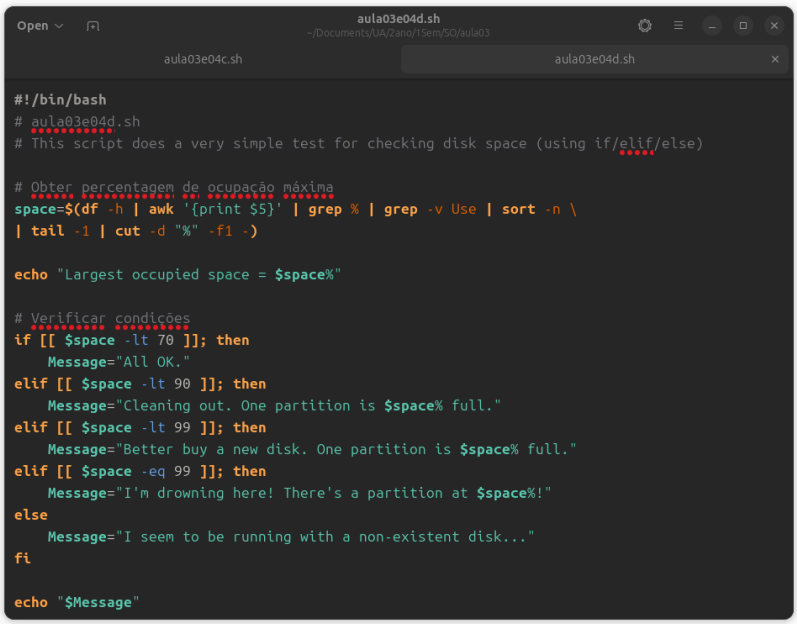
b)



c)



d)



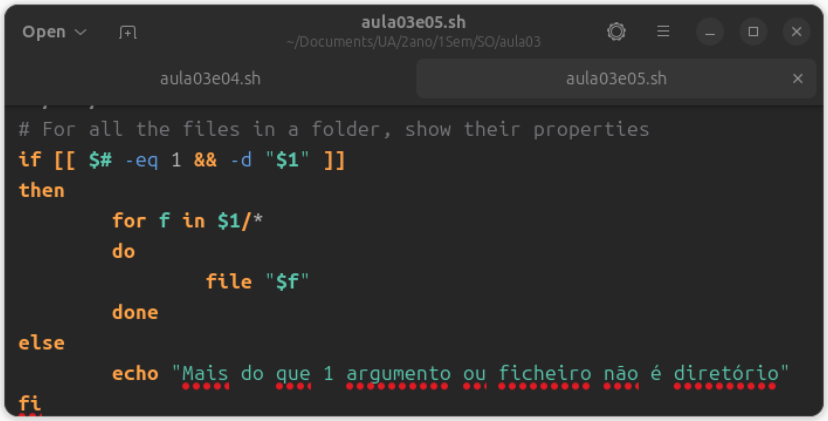
É mais fácil escrever condições númericas de forma mais direta a usar a estrutura if/else/fi. Porém a estutura do case é muito mais legível e deixa o script mais simples.

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.

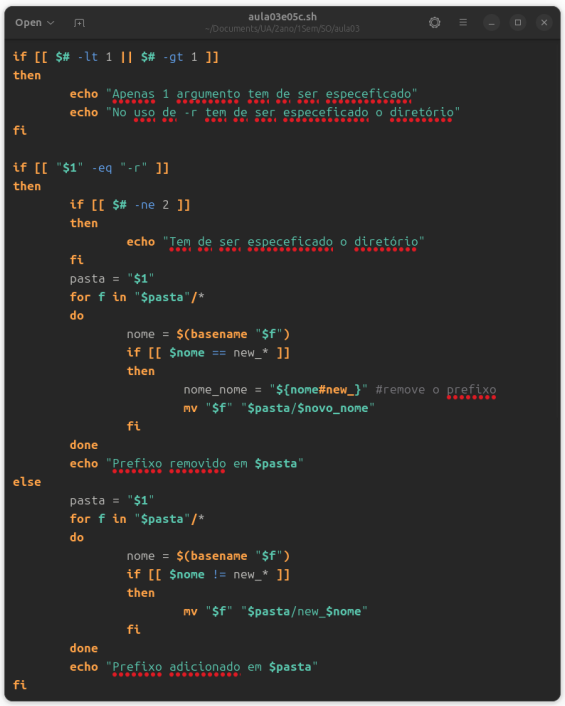
5.

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

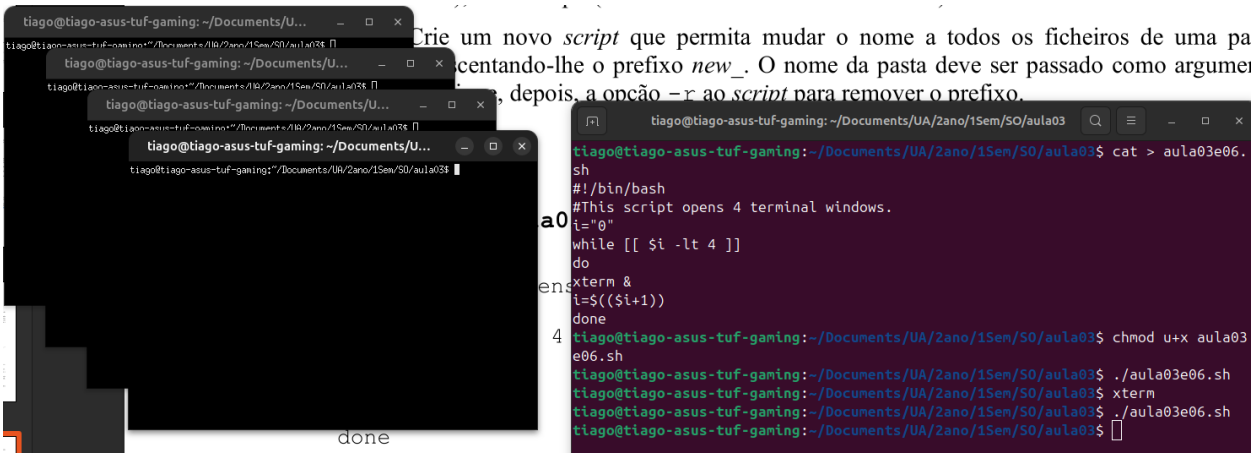
b)


Open
aula03e05.sh
~/Documents/UA/2ano/1Sem/SO/aula03
aula03e04.sh
aula03e05.sh
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)

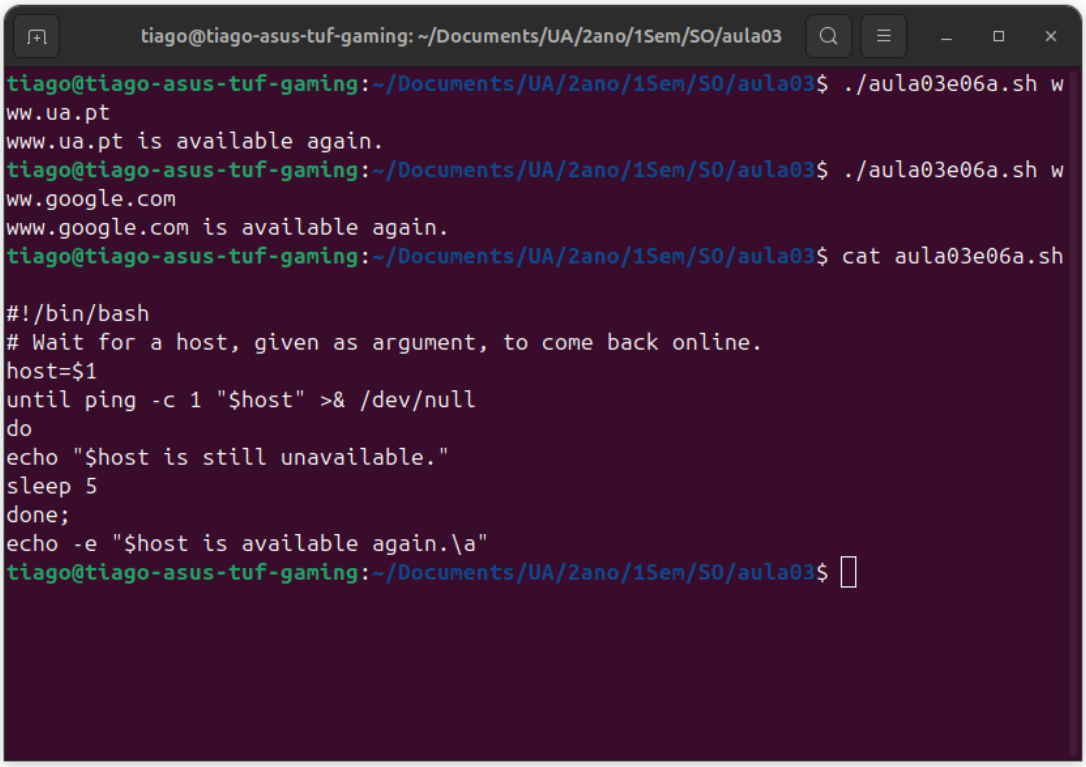

Open
aula03e05c.sh
~/Documents/UA/2ano/1Sem/SO/aula03
if [[$# -lt 1 || $# -gt 1]]
then
echo "Apenas 1 argumento tem de ser especeficado"
echo "No uso de -r tem de ser especeficado o diretório"
fi
if [["$1" -eq "-r"]]
then
if [[$# -ne 2]]
then
echo "Tem de ser especeficado o diretório"
fi
pasta = "$1"
for f in "$pasta"/*
do
nome = $(basename "$f")
if [[$nome == new_*]]
then
nome_nome = "${nome#new_}" #remove o prefixo
mv "$f" "$pasta/$novo_nome"
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.

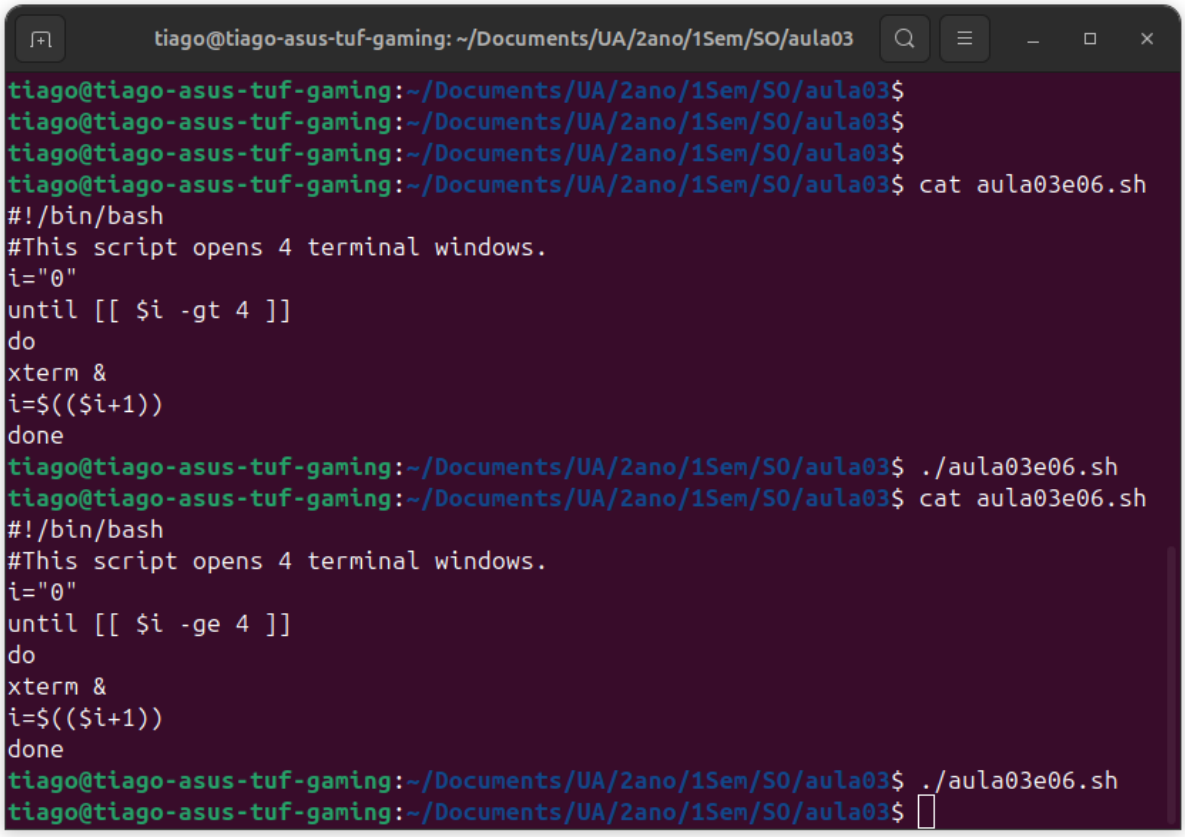
a)



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

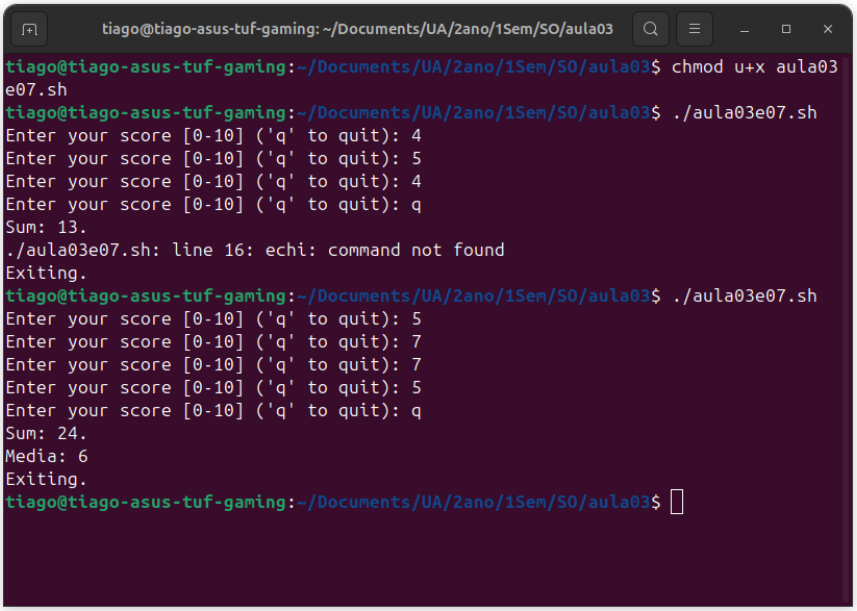
c)

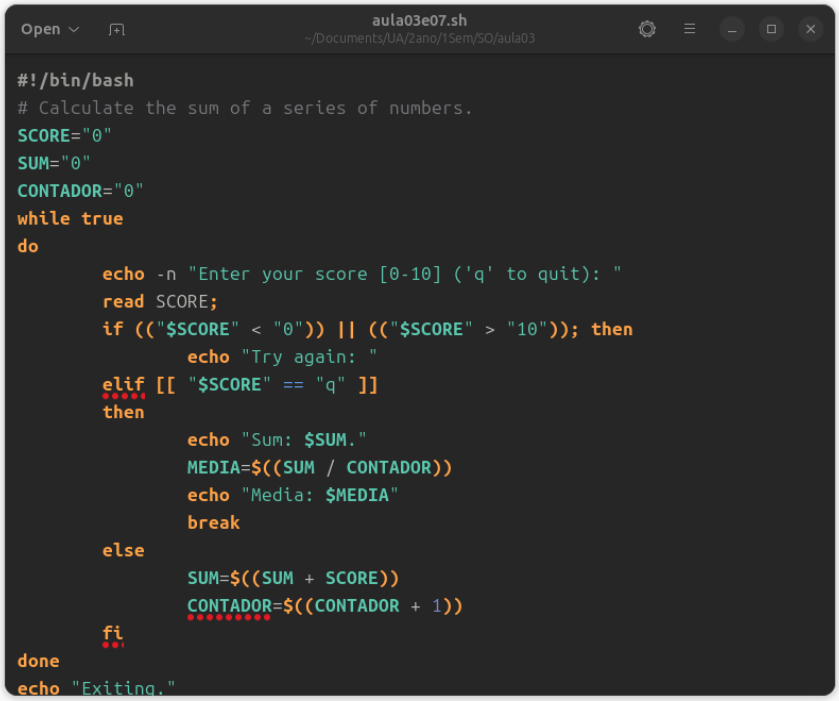


Para funcionar temos de usar o operador lógico oposto ao less than (-ls) que é o greater or equal then (-ge)

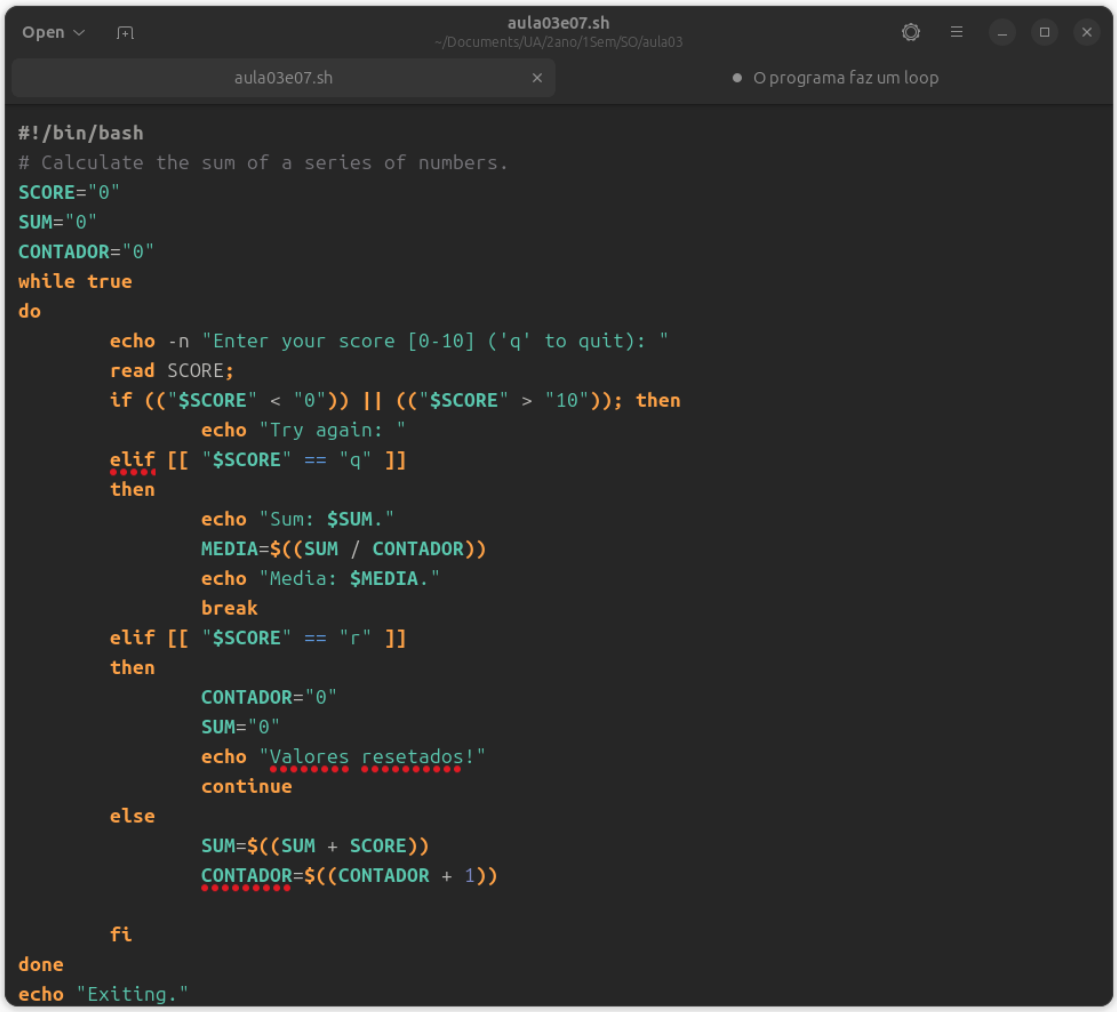
7. O programa faz um loop (do while) e enquanto a entrada não for "q" ele vai somando os valores inbutidos, e no final faz echo do resultado.

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."

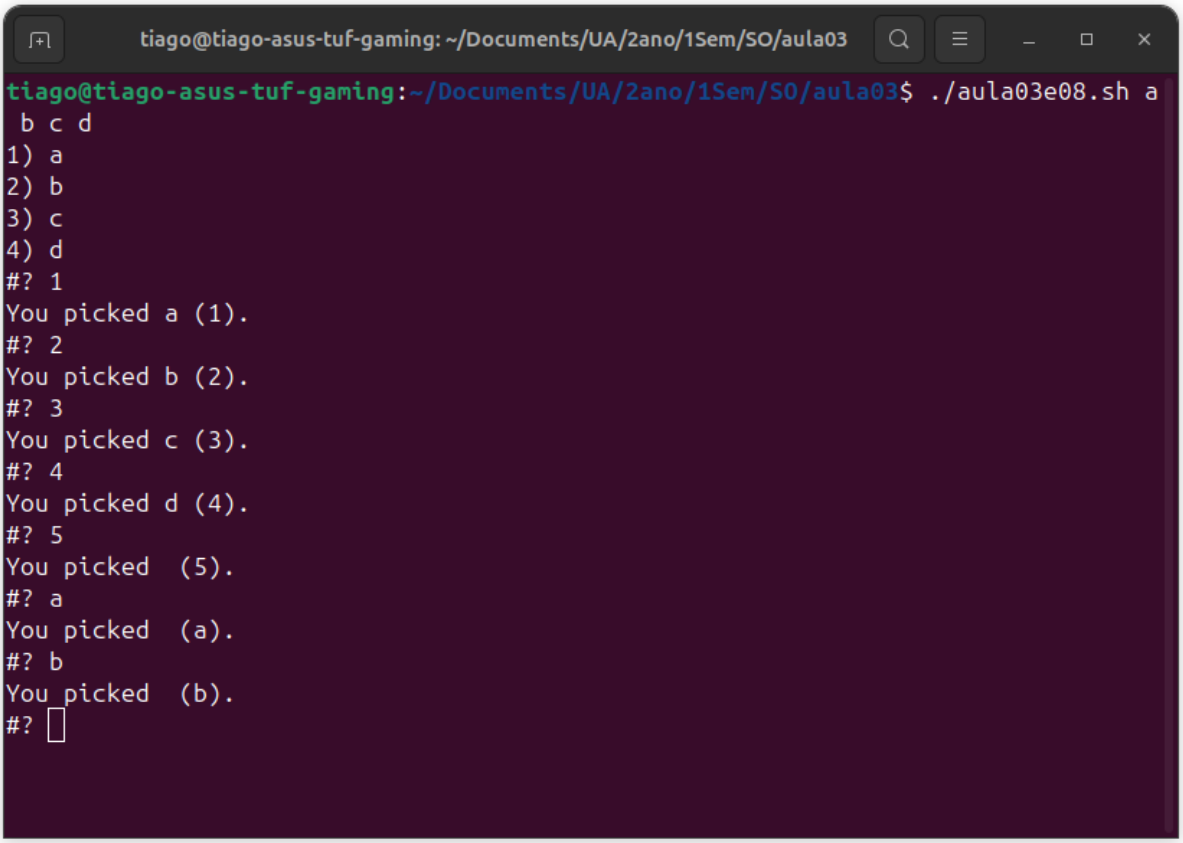
b)



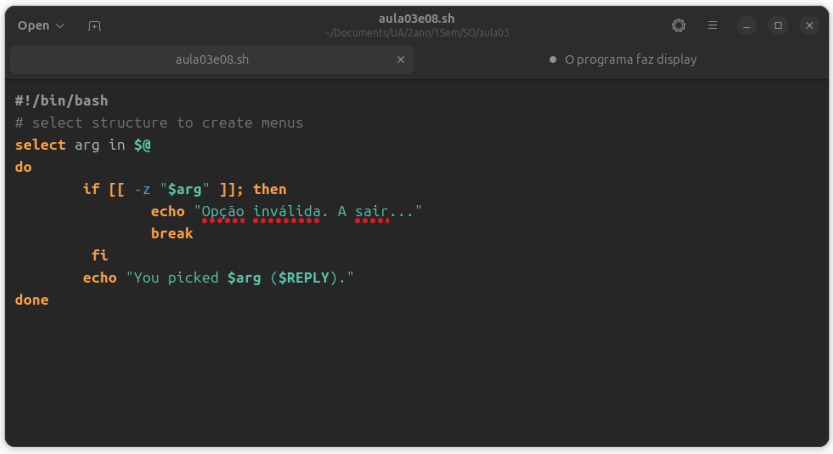
Fazemos uma nova condição se o valor do $SCORE for ”r”.

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



b)



Nota: “-z” para verificar se está vazio