



Introdução à Ciência da Computação – Lista 6 Shell script – parte 3

Nome: Larissa Rodrigues de Ávila

RA:2024.1.08.031

- 1) Crie um script chamado scriptaritmetico, com uma operação aritmética arbitrária usando pelo menos 4 variáveis, realizando uma operação de divisão cujo resultado não seja um número inteiro. Execute o script e mostre o resultado. Qual o recurso a ser utilizado caso você queira que o valor não inteiro apareça no resultado? Qual variável eu uso para isso?

```
scriptaritmetico.sh
1 #!/bin/bash
2 v1=1
3 v2=10
4 v3=2
5 v4=$((($v1 + $v2) / $v3)
6 echo "Resultado: $v4"
```

```
2024.1.08.031@suporte-OptiPlex-3050:~$ gedit scriptaritmetico.sh
2024.1.08.031@suporte-OptiPlex-3050:~$ chmod a+x scriptaritmetico.sh
2024.1.08.031@suporte-OptiPlex-3050:~$ ./scriptaritmetico.sh
Resultado: 5
2024.1.08.031@suporte-OptiPlex-3050:~$
```

Para que o valor não inteiro apareça no resultado, é necessário utilizar o recurso bc e uma variável scale e atribuir a ela a quantidade de casas decimais que você deseja no número.

- 2) Ponha em execução a calculadora bc. Mostre o uso da variável scale, exibindo um resultado de operação aritmética com 6 casas decimais.

```
2024.1.08.031@suporte-OptiPlex-3050:~$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
scale = 6
11 / 2
5.500000
quit
2024.1.08.031@suporte-OptiPlex-3050:~$
```

- 3) Crie um script simples chamado testebc, em que você utilize a calculadora bc dentro dele, envolvendo o uso de algumas variáveis e a operação de divisão, com o direcionamento via pipe. Execute o script, mostrando o resultado.

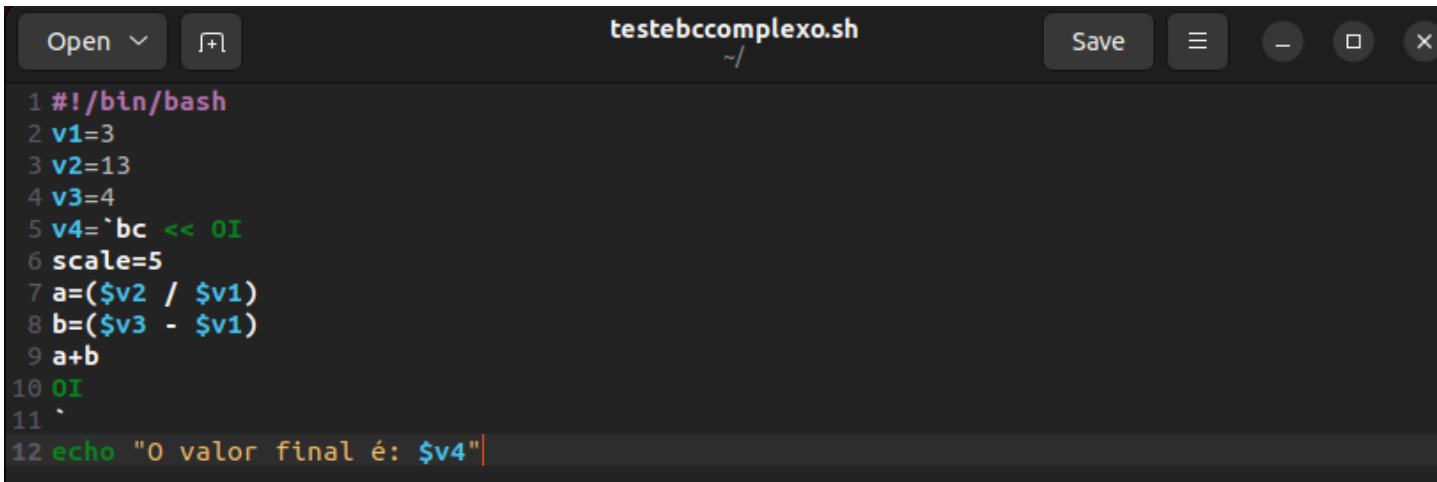
```
testebc.sh
1 #!/bin/bash
2 a=50
3 b=15
4 c=`echo "scale=3; $a / $b" | bc`
5 echo "O valor final é: $c"
```

```

2024.1.08.031@suporte-OptiPlex-3050:~$ gedit testebc.sh
2024.1.08.031@suporte-OptiPlex-3050:~$ ./testebc.sh
0 valor final é: 3.333

```

- 4) Crie um script chamado testebccomplexo, em que você utilize operações aritméticas diversas com a calculadora bc (pelo menos duas), armazenando os resultados em variáveis, como mostrado na aula. Neste caso, utilize a técnica de redirecionamento de entrada inline. Execute o script, mostrando o resultado.



```

1 #!/bin/bash
2 v1=3
3 v2=13
4 v3=4
5 v4=`bc << 0I
6 scale=5
7 a=($v2 / $v1)
8 b=($v3 - $v1)
9 a+b
10 0I
11 `
12 echo "0 valor final é: $v4"

```

```

2024.1.08.031@suporte-OptiPlex-3050:~$ gedit testebccomplexo.sh &
[1] 5960
2024.1.08.031@suporte-OptiPlex-3050:~$ chmod 755 testebccomplexo.sh
2024.1.08.031@suporte-OptiPlex-3050:~$ ./testebccomplexo.sh
0 valor final é: 5.33333

```

- 5) O que consiste o status de saída de um programa? Mostre um exemplo de execução de dois comandos (um com sucesso e outro desconhecido) e verifique esse status. Mostre em tela.

O status de saída de um programa é um número gerado no final de um programa que indica ao shell que o processamento terminou.

```

2024.1.08.031@suporte-OptiPlex-3050:~$ a=`echo "scale=2; 5 / 2" | bc`
2024.1.08.031@suporte-OptiPlex-3050:~$ echo "0 valor da divisão é $a"
0 valor da divisão é 2.50
2024.1.08.031@suporte-OptiPlex-3050:~$ echo $?
0

```

```

2024.1.08.031@suporte-OptiPlex-3050:~$ a=`echo "scale=2; 5 / 2 | bc`
bash: command substitution: line 1: unexpected EOF while looking for matching ``'
bash: command substitution: line 2: syntax error: unexpected end of file
2024.1.08.031@suporte-OptiPlex-3050:~$ echo $?
2

```

- 6) Qual a função do comando exit? Mostre um exemplo do uso do comando exit dentro de um script, mudando o valor padrão do status de saída. Mostre tanto o uso do exit exibindo um número qualquer até 255, quanto o valor de uma variável que você utilize no script. Execute o script e mostre o valor do status de saída em cada caso.

O comando exit permite especificar um status de saída quando o programa finaliza.

```
Open  [icon]  *testeexit.sh  Save  [menu]  [minus]  [square]  [x]

1 #!/bin/bash
2 a=32
3 b=4
4 c=$((a / b))
5 echo $c
6 exit 15
```

```
2024.1.08.031@suporte-OptiPlex-3050:~$ ./testeexit.sh
8
2024.1.08.031@suporte-OptiPlex-3050:~$ echo $?
15
```

```
Open  [icon]  testeexit.sh  Save  [menu]  [minus]  [square]  [x]

1 #!/bin/bash
2 a=32
3 b=4
4 c=$((a / b))
5 echo "finalizado"
6 exit $c
```

```
2024.1.08.031@suporte-OptiPlex-3050:~$ ./testeexit.sh
finalizado
2024.1.08.031@suporte-OptiPlex-3050:~$ echo $?
8
```

- 7) Crie um script simples envolvendo comandos condicionais if then else, para verificar a existência de um diretório específico no seu home. Primeiro procure um diretório inexistente, depois um diretório existente e exiba as mensagens específicas de acordo com o resultado. Execute o script e mostre em tela.

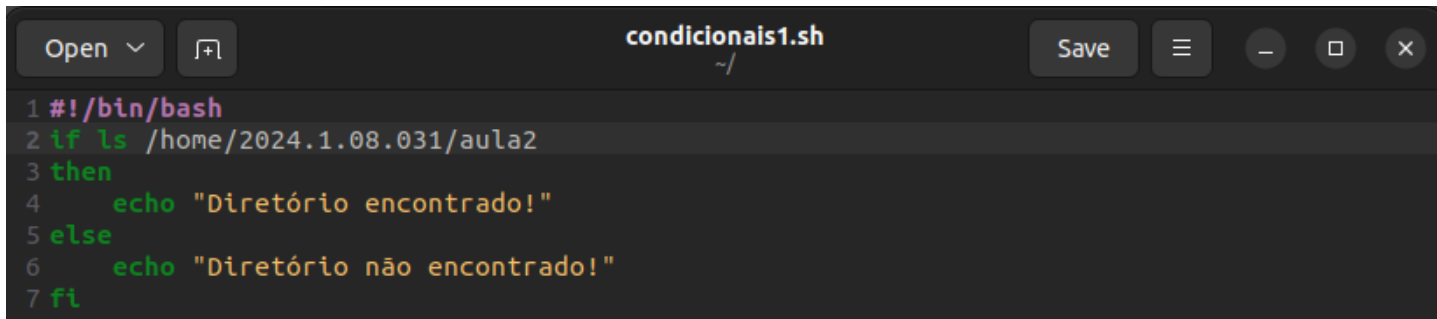
```
Open  [icon]  *condicionais1.sh  Save  [menu]  [minus]  [square]  [x]

1 #!/bin/bash
2 if ls /home/2024.1.08.031/aula1
3 then
4     echo "Diretório encontrado!"
5 else
6     echo "Diretório não encontrado!"
7 fi
```

```

2024.1.08.031@suporte-OptiPlex-3050:~$ ls
arq.txt          Downloads        NetBeansProjects  testeabc.sh
arquivo.sh       funcao          Pictures          testecrases.sh
aula1            icc             Public            testeexit.sh
condicionais1.sh intcc           scriptaritmetico.sh testevariaveisambiente.sh
Desktop          listadir.2904241147 snap              testevariaveis.sh
diretorio        listadir.2904241148 Templates         trabalho
doc.txt          listadir.2904241149 teste            Videos
Documents        Music           testebccomplexo.sh
2024.1.08.031@suporte-OptiPlex-3050:~$ ./condicionais1.sh
Diretório encontrado!

```



```

1 #!/bin/bash
2 if ls /home/2024.1.08.031/aula2
3 then
4     echo "Diretório encontrado!"
5 else
6     echo "Diretório não encontrado!"
7 fi

```

```

2024.1.08.031@suporte-OptiPlex-3050:~$ ./condicionais1.sh
ls: cannot access '/home/2024.1.08.031/aula2': No such file or directory
Diretório não encontrado!

```

- 8) Crie um script envolvendo várias condicionais usando a estrutura if then elif else, fazendo duas operações aritméticas arbitrárias, verificando o valor das variáveis que armazenam essa operação, checando se o valor da primeira é maior, menor ou igual ao valor da segunda. Execute o script e mostre o resultado em tela.



```

Abrir  +
testecondicoes.sh

#!/bin/bash
var1=6
var2=2
var3=3
var4=$(( $var1 * $var2 ))
var5=$(( $var3 * $var2 * $var2 ))
if [ $var4 -gt $var5 ]
then
    echo "A variável 1 é maior que a variável 2"
elif [ $var4 -eq $var5 ]
then
    echo "As variáveis tem valores iguais"
else
    echo "A variável 2 é maior que a variável 1"
fi

```

```
larissa-avila@larissa-avila:~$ chmod 755 testecondicoes.sh
larissa-avila@larissa-avila:~$ ./testecondicoes.sh
As variáveis tem valores iguais
```

- 9) Crie um script envolvendo condicionais usando a estrutura if then else, criando duas variáveis string arbitrárias e verificando seus valores, checando se o conteúdo das variáveis é igual. Execute o script e mostre o resultado em tela.

Abrir ▾ testecondicoes.sh

```
#!/bin/bash
var1=larissa
var2=nayene
if [ $var1 = $var2 ]
then
    echo "As variáveis são iguais"
else
    echo "As variáveis são diferentes"
fi
```

```
larissa-avila@larissa-avila:~$ ./testecondicoes.sh
As variáveis são diferentes
```

- 10) Crie um script envolvendo condicionais usando a estrutura if then else, criando uma string com um conteúdo, verificando se seu valor é "fruta". Execute o script e mostre o resultado em tela.

```
1 #!/bin/bash
2 var1=fruta
3 if [ $var1 = "fruta" ]
4 then
5     echo "A string armazenada é fruta"
6 else
7     echo "A string armazenada não é fruta"
8 fi
```

```
larissa-avila@larissa-avila:~$ ./testecondicoes.sh
A string armazenada é fruta
```

- 11) Crie um script envolvendo condicionais usando a estrutura if then else, criando duas strings, uma vazia, outra com conteúdo e verificando estes resultados (se tem conteúdo em ambos os casos).

```

1 #!/bin/bash
2 var1=hello
3 var2=''
4 if [ -z $var1 ]
5 then
6     echo "A variável 1 está vazia"
7 else
8     echo "A variável 1 não está vazia, ela contém o valor $var1"
9 fi
10
11 if [ -z $var2 ]
12 then
13     echo "A variável 2 está vazia"
14 else
15     echo "A variável 2 não está vazia, ela contém o valor $var2"
16 fi

```

```

larissa-avila@larissa-avila:~$ ./testecondicoes.sh
A variável 1 não está vazia, ela contém o valor hello
A variável 2 está vazia

```

12) Cite 5 opções de comparações envolvendo arquivos. Escolha uma das opções e crie um script envolvendo essa opção.

-e arquivo: verifica se o arquivo existe

-d arquivo: verifica se o arquivo existe e se é um diretório

arquivo1 -nt arquivo2: verifica se o arquivo1 é mais novo que o arquivo2

arquivo1 -ot arquivo2: verifica se o arquivo1 é mais antigo que o arquivo2

-s arquivo: verifica se o arquivo existe e não está vazio

```

1 #!/bin/bash
2 arquivo1=/home/larissa-avila/testecondicoes.sh
3 arquivo2=/home/larissa-avila/testearquivos.txt
4 if [ arquivo1 -nt arquivo2 ]
5 then
6     echo "O arquivo testecondicoes é mais novo que o arquivo testearquivos"
7 else
8     echo "O arquivo testearquivos é mais novo que o arquivo testecondicoes"
9 fi

```

```

larissa-avila@larissa-avila:~$ ./testecondicoes.sh
O arquivo testearquivos é mais novo que o arquivo testecondicoes

```