MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas
UNIFAL-MG
Rua Gabriel Monteiro da Silva, 700 – Centro. Alfenas/MG. CEP: 37130-001

Introdução à Ciência da Computação – Lista 7
Shell script – parte 4

Nome: Gabriela Mazon Rabello de Souza **RA:2025.1.08.006**

1) Crie um script chamado escrevenome, faça com que a saída desse script seja seu nome completo. Não utilize o comando chmod. Depois crie um script chamado testecompara, utilize o operador AND e verifique se o usuário logado tem permissão r e x sobre o script escrevenome. Mostre o resultado da saída.

```
#!/bin/bash
#exercício 1
echo "Gabriela Mazon Rabello de Souza"
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit escrevenome.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash escrevenome.sh
Gabriela Mazon Rabello de Souza
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit escrevenome.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit tesecompara.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash tesecompara.sh
O usuário 2025.1.08.006 não tem permissão para alterar o arquivo
```

```
#!/bin/bash
#exercício 1 teste compara
if [ -r escrevenome.sh ] && [ -x escrevenome.sh ]
then
echo "O usuário $USER tem permissão de leitura e exercução sobre
 arquivo"
else
echo "O usuário $USER não tem permissão para alterar o arquivo"
fi
```

2) Crie um script chamado frutascase. Com base no valor da variável fruta mostre uma breve descrição da fruta. Faça com 5 frutas. Exemplo: fruta=uva, echo “A uva é o fruto da videira ou parreira, uma planta da família Vitaceae. É originária da Ásia e uma das frutas mais antigas utilizadas na alimentação humana. Existem mais de 60 mil variedades da fruta. A cor, o sabor e o tamanho variam de acordo com cada espécie. A uva também é classificada quanto ao destino de produção, de mesa ou para vinicultura. Pode ser consumida in natura ou usada na preparação de doce, vinho, passas, musses, geléias, tortas, gelatinas, sucos.”

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit frutascase.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash frutascase.sh manga
A manga é tropical e cheia de vitaminas.
2025.1.08.006@suporte-OptiPlex-3050:~$ bash frutascase.sh goiaba
Fruta não reconhecida.
```

```bash
#!/bin/bash
#Exercicio 2 script frutascase
fruta=$1

case $fruta in
uva)
    echo "A uva é uma das frutas mais antigas..."
    ;;
banana)
    echo "A banana é rica em potássio..."
    ;;
maçã)
    echo "A maçã é uma fruta rica em fibras..."
    ;;
manga)
    echo "A manga é tropical e cheia de vitaminas."
    ;;
abacaxi)
    echo "O abacaxi é uma fruta tropical ácida."
    ;;
*)
    echo "Fruta não reconhecida."
    ;;
esac
```

3) Cite, explique e faça um script simples para cada estrutura de repetição do shell bash. Use sua criatividade para os scripts.

-For: O comando for permite criar um loop que itera através de uma série de valores, cada interação executa um conjunto definido de comandos usando um dos valores da lista.

```bash
#!/bn/bash
#Exercicio 3 - for
for comida in arroz fejão massas pizza "pão de queijo"
do
echo "Eu gosto de $comida"
done
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ bash for.sh
Eu gosto de arroz
Eu gosto de fejão
Eu gosto de massas
Eu gosto de pizza
Eu gosto de pão de queijo
```

-While: O comando while permite definir um comando a testar e então iterar por um conjunto de comandos enquanto o comando definido de teste retornar status de saída zero.

```bash
#!/bin/bash
#exercício 3 while
contador=5
while [ $contador -gt 0 ]
do
echo "Contando: $contador"
contador=$((contador - 1))
done
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit while.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash while.sh
Contando: 5
Contando: 4
Contando: 3
Contando: 2
Contando: 1
```

-Until:O comando until opera de forma oposta ao comando while.É necessário especificar um comando de teste que retorne um status de saída diferente de zero para que o bloco de comandos listado no loop seja executado.

```
#!/bin/bash
#Exercício 3 until
valor=1
until [ $valor -gt 10 ]
do
echo "valor: $valor"
valor=$((valor*2))
done
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit until.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash until.sh
valor: 1
valor: 2
valor: 4
valor: 8
```

4) Explique o que é IFS e faça um script diferente do que foi visto em aula. Use sua criatividade.

IFS é uma variável de ambiente (Internal Field Separator), a qual define uma lista de caracteres que o shell bash usa como separadores de campos.

```
#!/bin/bash
#Exercício 4 IFS
lista="arroz:feijão:batata:carne:refrigerante"
IFS=":"
for item in $lista
do
echo "Item: $item"
done
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit ifs.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash ifs.sh
Item: arroz
Item: feijão
Item: batata
Item: carne
Item: refrigerante
```

5) Crie um script for no estilo C que mostre na tela os números de 50 a 20.

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit forc.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash forc.sh
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
```

```
1 #!/bin/bash
2 #Exercício 5
3 for ((i=50; i>=20; i--))
4 do
5 echo "$i"
6 done
```

6) Desenvolva um script que receba um parâmetro e verifique se o valor está entre 0 e 10. Caso sim mostre o triplo do valor. Caso ele esteja entre 10 e 20 mostre o dobro. Caso não esteja nos anteriores apresente uma mensagem.

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit valores.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash valores.sh 4
Triplo de 4: 12
2025.1.08.006@suporte-OptiPlex-3050:~$ bash valores.sh 15
Dobro de 15: 30
2025.1.08.006@suporte-OptiPlex-3050:~$ bash valores.sh 25
 Valor fora da faixa.
```

```bash
#!/bin/bash
#Exercício 6
valor=$1
if [ $valor -ge 0 ] && [ $valor -le 10 ]; then
echo "Triplo de $valor: $((valor * 3))"
elif [ $valor -gt 10 ] && [ $valor -le 20 ];then
echo "Dobro de $valor: $((valor * 2))"
else
echo " Valor fora da faixa."
fi
```

7) Explique o que é $# e faça um script diferente do que foi visto em aula. Faça com dois parâmetros. Use sua criatividade.

A variável especial $# contém o número de parâmetros de linhas de comando fornecidos ao rodar o script. Podemos usá-la para verificar se o usuário digitou o número de parâmetros necessários para rodar o programa corretamente.

```bash
#!/bin/bash
#Exercício 7
if [ $# -ne 2 ];then
echo "ERRO: forneça 2 números"
exit 1
fi
soma=$(( $1 + $2 ))
produto=$(( $1 * $2 ))
echo "Soma: $soma"
echo "Produto: $produto"
```

```
2025.1.08.006@suporte-OptiPlex-3050:~$ gedit parametros.sh
2025.1.08.006@suporte-OptiPlex-3050:~$ bash parametros.sh 5 3
Soma: 8
Produto: 15
2025.1.08.006@suporte-OptiPlex-3050:~$ bash parametros.sh 7
ERRO: forneça 2 números
2025.1.08.006@suporte-OptiPlex-3050:~$
```