



Rua Gabriel Monteiro da Silva, 700 - Centro. Alfenas/MG. CEP: 37130-001

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

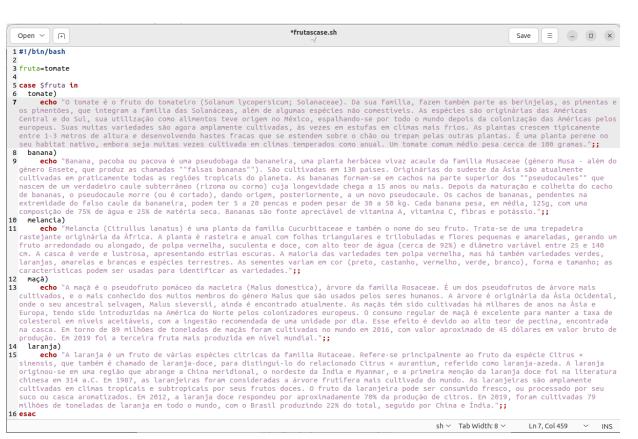
Nome: Gabriel Ribeiro RA: 2024.1.08.010

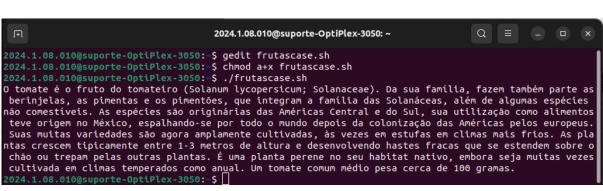
1) Crie um script chamado escrevenome, faça com que a saída desse script seja seu nome completo. Não utilize o comando chmod. Depois crie um script chamado testecompara, utilize o operador AND e verifique se o usuário logado tem permissão r e x sobre o script escrevenome. Mostre o resultado da saída.

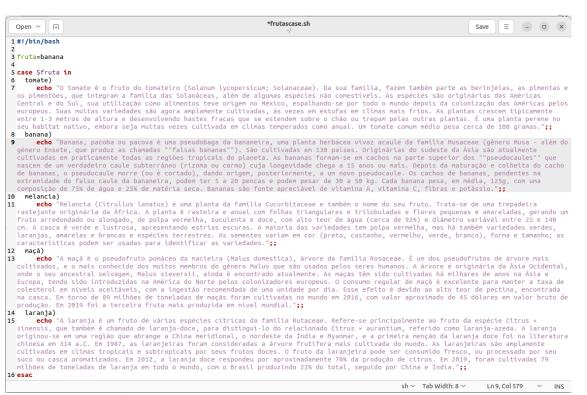


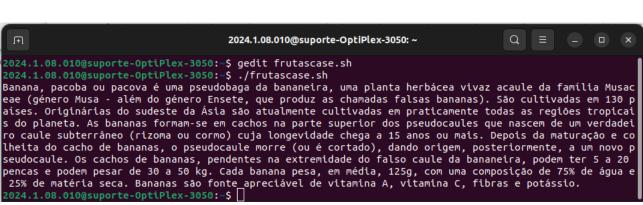


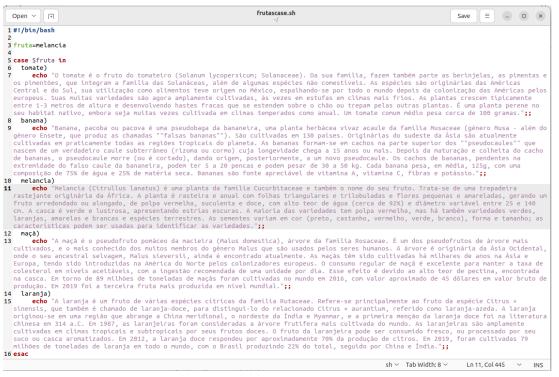
2) Crie um script chamado frutascase. Com base no valor da variável fruta mostre uma breve descrição da fruta. Faça com 5 frutas. Exemplo: fruta=uva, echo "A uva é o fruto da videira ou parreira, uma planta da família Vitaceae. É originária da Ásia e uma das frutas mais antigas utilizadas na alimentação 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."

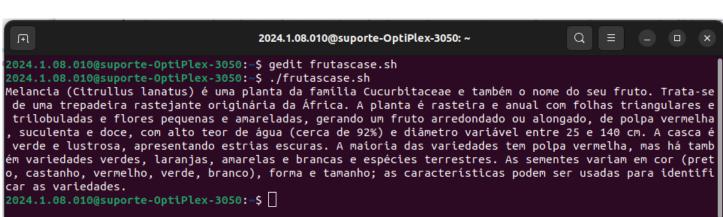


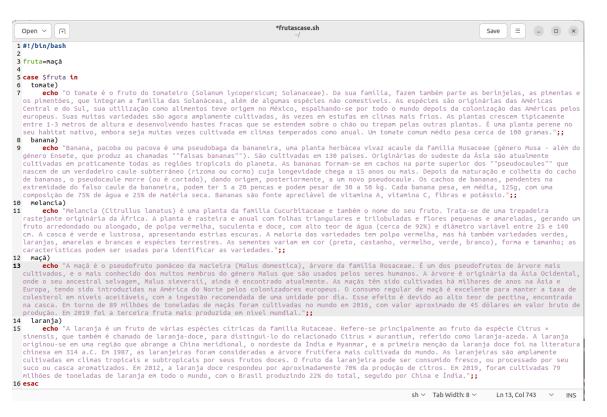


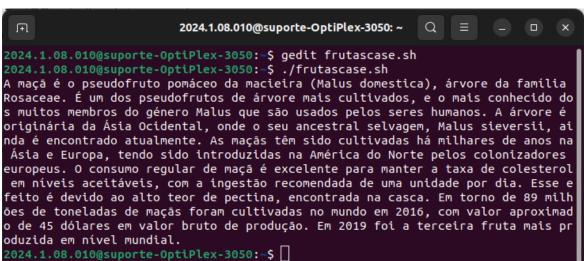


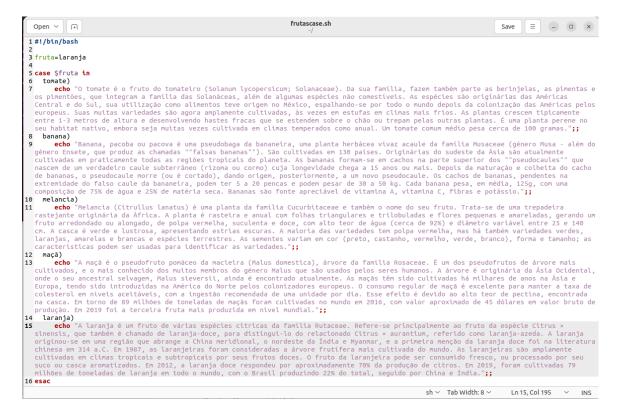


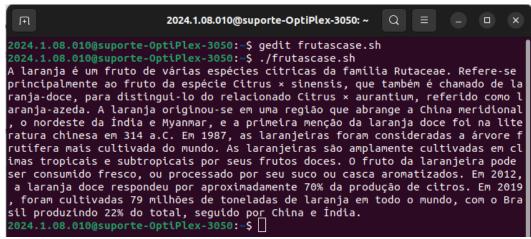












3) Cite, explique e faça um script simples para cada estrutura de repetição do shell bash. Use

sua criatividade para os scripts.

Comando FOR: É usado para repetir um bloco de comandos que é controlado por um contador, sendo incrementado em cada ciclo terminado.

```
2024.1.08.010@suporte-OptiPlex-3050: ~
 ſŦ
                                          Q.
                                                         ×
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit scriptfor.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./scriptfor.sh
Detonação da bomba em: 9
Detonação da bomba em: 8
Detonação da bomba em: 7
Detonação da bomba em: 6
Detonação da bomba em: 5
Detonação da bomba em: 4
Detonação da bomba em: 3
Detonação da bomba em: 2
Detonação da bomba em: 1
Detonação da bomba em: 0
BOOMMM!
2024.1.08.010@suporte-OptiPlex-3050:~$
```

Comando WHILE: É um comando de repetição, no qual executa um trecho de códigos repetidas vezes enquanto a condição dele retornar valor 0.

```
scriptwhile.sh
  Open ~
            (<del>+</del>)
                                     Save
                                                  _ D X
 1 #!/bin/bash
 3 echo "Numeros com olhos"
 4 contagem=0
 6 while [ $contagem -le 10 ]
 7 do
 8
 9
      echo "$contagem + : = :$contagem"
      contagem=$[$contagem +1]
10
11 done
```

```
2024.1.08.010@suporte-OptiPlex-3050: ~
                                          Q
                                               \equiv
 Æ
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit scriptwhile.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod a+x scriptwhile.s
h
2024.1.08.010@suporte-OptiPlex-3050:~$ ./scriptwhile.sh
Numeros com olhos
0 + : = :0
 +:=:1
 +:=:2
   : = :3
   : = :4
    : = :5
8 + : = :8
9 + : = :9
10 + : = :10
2024.1.08.010@suporte-OptiPlex-3050:~$
```

Comando UNTIL: É o inverso do comando while, executando comandos enquanto o teste

retorna valores diferentes de 0.

```
scriptuntil...
  Open ~
                                                  Save
 1 #!/bin/bash
2
3 valor=7
4 echo "Multiplos de $valor"
5 mult=10
7 until [ $mult -eq 0 ]
8 do
9
     resultado=$[$valor*$mult]
     echo "$valor * $mult = $resultado"
     mult=$[$mult -1]
11
12 done
```

```
2024.1.08.010@suporte-OptiPlex-3050: ~
                                          Q
                                                         2024.1.08.010@suporte-OptiPlex-3050:~$ gedit scriptuntil.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./scriptuntil.sh
Multiplos de 7
7 * 10 = 70
 * 9 = 63
  * 8 = 56
     = 49
  * 6 = 42
   5 = 35
   4 = 28
  * 3 = 21
  * 2 = 14
2024.1.08.010@suporte-OptiPlex-3050:~$
```

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

IFS (Internal Field Separator) é uma lista de caracteres que o shell bash considera como separadores de campos. Sendo eles o espaço, tabulação e newline.

```
Open > If ifs.sh Save = - - x

1 #!/bin/bash
2
3 arquivo=/home/2024.1.08.010/Documents/motivacao.txt
4 IFSOLD=$IFS
5 IFS=$'\n'
6
7 for frase in `cat $arquivo`
8 do
9 echo $frase
10 done
11 IFS=$IFSOLD
```

```
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit ifs.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod a+x ifs.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod a+x ifs.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./ifs.sh
Frases motivacionais:
É importante agradecer pelo hoje sem nunca desistir do amanhā!
A ideia é nunca perder a motivação para subir cada degrau rumo ao sucesso, por mais difícil que seja.
E também não esquecer de agradecer pelas lições aprendidas em cada etapa da jornada.
Quem cultiva gratidão é capaz de realizar sonhos que parecem inalcançáveis!
Quem cultiva gratidão é capaz de realizar sonhos que parecem inalcançáveis!
2024.1.08.010@suporte-OptiPlex-3050:~$
```

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

```
Open > forc.sh Save = - x

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

```
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit forc.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod a+x forc.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./forc.sh
```

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.

```
parametro...
  Open ~
                                Save
                                        \equiv
                                                   ×
 1 #!/bin/bash
 3 if [ $1 -ge 0 ] && [ $1 -le 10 ]
 4 then
     resultado=$[$1*3]
 5
     echo "O triplo de $1 é = $resultado"
 7 elif [ $1 -gt 10 ] && [ $1 -le 20 ]
 8 then
 9
     resultado=$[$1*2]
     echo "O dobro de $1 é = $resultado"
10
11 else
     echo "O valor inserido não está entre O e 20."
12
13 fi
```

```
2024.1.08.010@suporte-OptiPlex-3050: ~
                                             Q
                                                  \equiv
 Ŧ
                                                             ×
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit parametroif.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod 755 parametroif.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroif.sh
./parametroif.sh: line 3: [: -ge: unary operator expected
./parametroif.sh: line 7: [: -gt: unary operator expected
O valor inserido não está entre O e 20.
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroif.sh 4
0 \text{ triplo de } 4 \text{ \'e} = 12
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroif.sh 13
O dobro de 13 é = 26
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroif.sh 40
O valor inserido não está entre O e 20.
2024.1.08.010@suporte-OptiPlex-3050:~$
```

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

parâmetros. Use sua criatividade.

\$# 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.

```
parametroesp...
  Open ~
            \Box
                                   Save
                                                     ×
 1 #!/bin/bash
3 if [ $# -lt 1 ]
4 then
 5 echo "Precisa fornecer pelo menos 1 argumento!"
7 echo "Número de argumentos passados: $#"
8 i=0
9 for argumento in $*
10 do
11 i=$(($i+1))
12 echo "Argumento $i passado: $argumento"
```

```
2024.1.08.010@suporte-OptiPlex-3050: ~
                                                                Q
2024.1.08.010@suporte-OptiPlex-3050:~$ gedit parametroespecial.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ chmod 755 parametroespecial.sh
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroespecial.sh
Precisa fornecer pelo menos 1 argumento!
Número de argumentos passados: 0
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroespecial.sh 4
Número de argumentos passados: 1
Argumento 1 passado: 4
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroespecial.sh 9
2024.1.08.010@suporte-OptiPlex-3050:~$ ./parametroespecial.sh 9 jose
Número de argumentos passados: 2
Argumento 1 passado: 9
Argumento 2 passado: jose
2024.1.08.010@suporte-OptiPlex-3050:~$
```