



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

Nome: Pedro Augusto de Souza Finochio RA: 2024.1.08.020

- 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:

```
Open  [icon] escrevenome.sh ~/ Save [icon] [icon] [icon] [icon]
1 #!/bin/bash
2
3 echo "Pedro Augusto de Souza Finochio"
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash escrevenome.sh
Pedro Augusto de Souza Finochio
2024.1.08.020@suporte-OptiPlex-3050:~$
```

```
Open  [icon] testecompara.sh ~/ Save [icon] [icon] [icon] [icon]
1 #!/bin/bash
2
3 if [ $USER = Pedro ] && [ -r $HOME/.escrevenome ] && [ -x $HOME/.escrevenome ]
4 then
5     echo "O usuário tem permissões de leitura e execução do arquivo"
6 else
7     echo "O usuário não tem permissões de leitura e execução do arquivo"
8 fi
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testecompara.sh
O usuário não tem permissões de leitura e execução do arquivo
2024.1.08.020@suporte-OptiPlex-3050:~$
```

- 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.”

```
Open  ▾  frutascase.sh  Save  ⋮  _  □  ×

1 #!/bin/bash
2
3 fruta="mexerica"
4
5 if [ "$fruta" = "mexerica" ]; then
6     echo "A mexerica, também conhecida como o robson, é uma pequena árvore cítrica com fruta
   parecida com outras laranjas."
7 elif [ "$fruta" = "banana" ]; then
8     echo "A banana pode ser consumida in natura, além de ser empregada na fabricação de pratos,
   tortas, doces, sorvetes, etc. A fruta é consumida em larga escala nos Estados Unidos e Europa. A
   Índia é a maior produtora de bananas do mundo, seguida pelo Brasil, China, Equador, Filipinas,
   Indonésia, Costa Rica e México."
9 elif [ "$fruta" = "laranja" ]; then
10    echo "A laranja é uma das frutas mais populares do mundo. Ela pertence ao grupo das frutas
   cítricas, mas sua origem é um mistério. Acredita-se que as primeiras laranjas foram cultivadas
   no leste da Ásia há milhares de anos, porém elas são usadas e amadas no mundo todo. No Brasil,
   por exemplo, a laranja-pera é a mais conhecida e consumida."
11 elif [ "$fruta" = "abacaxi" ]; then
12    echo "O benefício de comer abacaxi à noite está relacionado ao sono. Isso porque este
   alimento contém nutrientes os quais estimulam a produção de serotonina, além de diminuir a
   ansiedade. Desta forma, o consumo do abacaxi à noite pode trazer uma noite mais tranquila e uma
   boa noite de sono."
13 elif [ "$fruta" = "pera" ]; then
14    echo "Pode comer pera com casca? Sim. A casca da pera contém polifenóis, um tipo de
   antioxidante poderoso. Consumir a casca da fruta pode oferecer seis vezes mais polifenóis do que
   sua "carne".
15 else
16    echo "Nenhuma fruta encontrada."
17 fi
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash frutascase.sh
A mexerica, também conhecida como o robson, é uma pequena árvore cítrica com fruta parecida com outras laranjas.
2024.1.08.020@suporte-OptiPlex-3050:~$
```

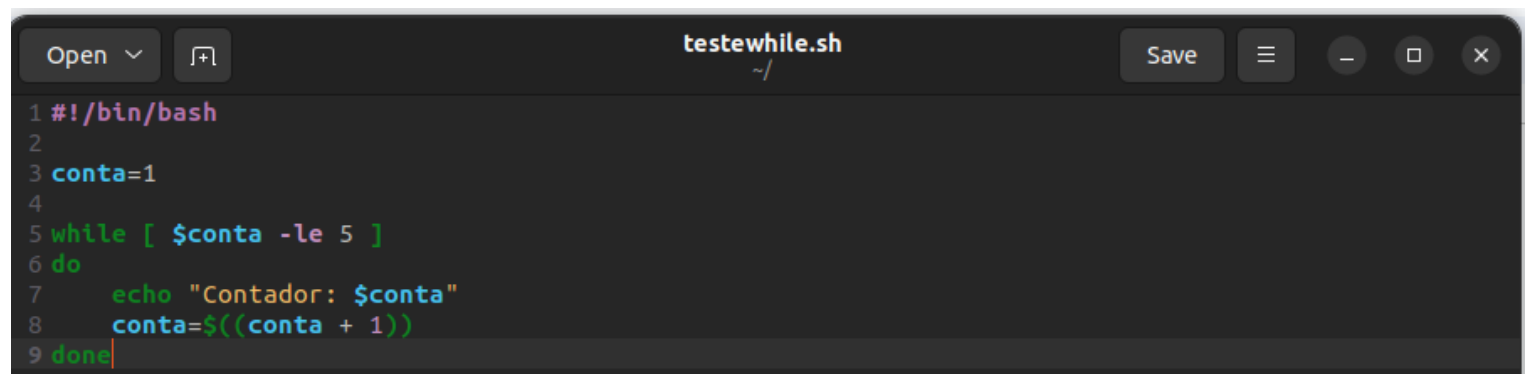
- 3) Cite, explique e faça um script simples para cada estrutura de repetição do shell bash. Use sua criatividade para os scripts: O loop for é usado para iterar sobre uma lista de itens:

```
Open  ▾  testefor.sh  Save  ⋮  _  □  ×

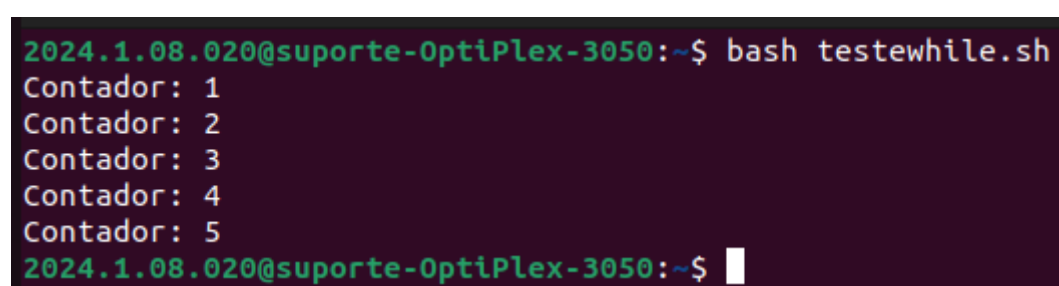
1 #!/bin/bash
2
3 for dia in Segunda Terça Quarta Quinta Sexta Sábado Domingo
4 do
5     echo "Hoje é $dia"
6 done
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testefor.sh
Hoje é Segunda
Hoje é Terça
Hoje é Quarta
Hoje é Quinta
Hoje é Sexta
Hoje é Sábado
Hoje é Domingo
2024.1.08.020@suporte-OptiPlex-3050:~$
```

O loop while é usado para executar um bloco de código repetidamente enquanto uma condição específica for verdadeira:

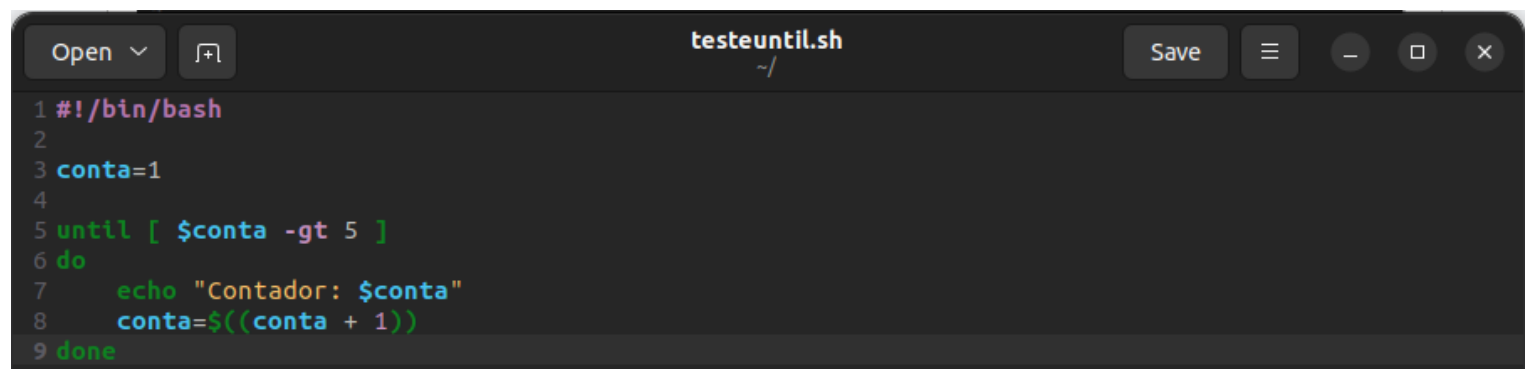
A screenshot of a code editor window titled 'testewhile.sh'. The editor has a dark theme and shows a Bash script. The script starts with a shebang line, followed by a variable assignment, a while loop condition, and a do-while loop body that prints the counter and increments it. The script ends with a 'done' keyword.

```
1 #!/bin/bash
2
3 conta=1
4
5 while [ $conta -le 5 ]
6 do
7     echo "Contador: $conta"
8     conta=$((conta + 1))
9 done
```

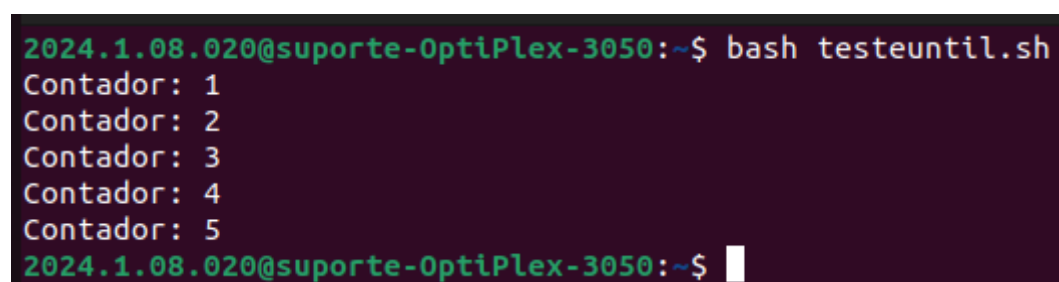
A screenshot of a terminal window showing the execution of the 'testewhile.sh' script. The prompt is '2024.1.08.020@suporte-OptiPlex-3050:~\$'. The script runs and outputs five lines, each showing 'Contador:' followed by a number from 1 to 5. The terminal prompt returns after the last line.

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testewhile.sh
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
2024.1.08.020@suporte-OptiPlex-3050:~$
```

O comando 'until' é uma estrutura de controle de repetição em Bash que é muito semelhante ao comando 'while'. A diferença principal é que o comando 'until' executa um bloco de código repetidamente enquanto uma condição é falsa, ao contrário do 'while', que executa enquanto a condição é verdadeira:

A screenshot of a code editor window titled 'testeuntil.sh'. The editor has a dark theme and shows a Bash script. The script starts with a shebang line, followed by a variable assignment, an until loop condition, and a do-until loop body that prints the counter and increments it. The script ends with a 'done' keyword.

```
1 #!/bin/bash
2
3 conta=1
4
5 until [ $conta -gt 5 ]
6 do
7     echo "Contador: $conta"
8     conta=$((conta + 1))
9 done
```

A screenshot of a terminal window showing the execution of the 'testeuntil.sh' script. The prompt is '2024.1.08.020@suporte-OptiPlex-3050:~\$'. The script runs and outputs five lines, each showing 'Contador:' followed by a number from 1 to 5. The terminal prompt returns after the last line.

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testeuntil.sh
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
2024.1.08.020@suporte-OptiPlex-3050:~$
```

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

IFS é uma abreviação para "Internal Field Separator" (Separador Interno de Campos) e é uma variável de ambiente importante em shell scripting. Ela determina como o shell divide as strings em palavras (ou campos) quando são lidas a partir de variáveis ou de entrada padrão:

```
testelIFS.sh
~/
Open  Save  -  □  ×
1 #!/bin/bash
2
3 ditado="Não deixe para amanhã , o que voce pode fazer hoje"
4
5 SalvarIFS=$IFS
6
7 IFS=" "
8
9 for sentenca in $ditado
10 do
11     echo "Sentença: $sentenca"
12 done
13
14 IFS=$SalvarIFS
```

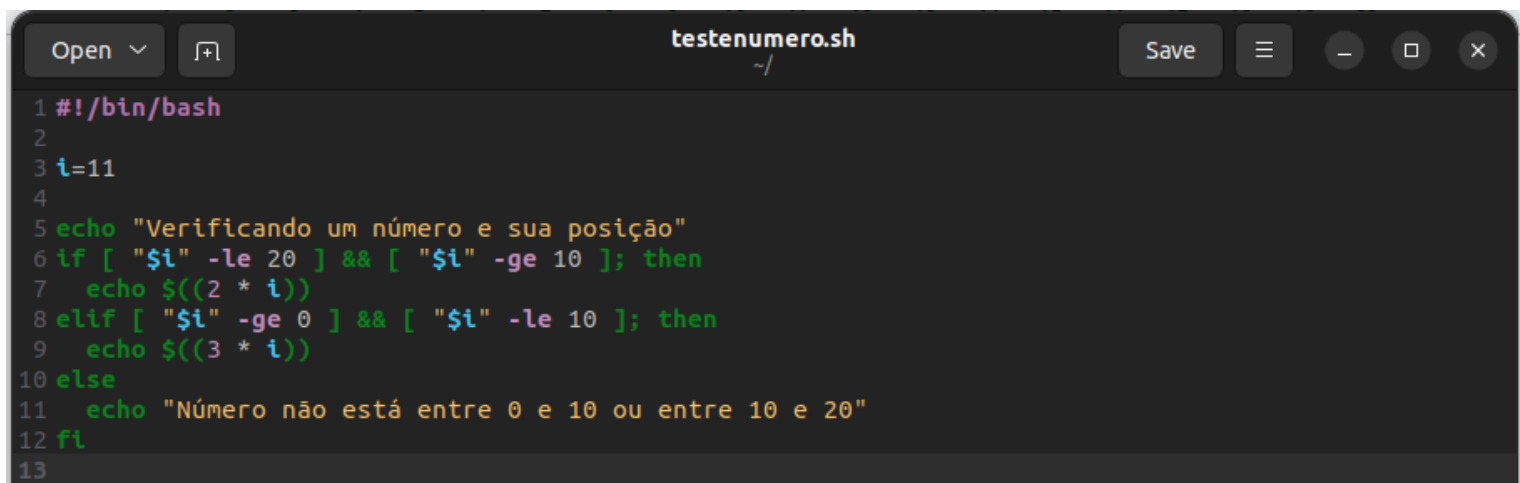
```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testeIFS.sh
Sentença: Não
Sentença: deixe
Sentença: para
Sentença: amanhã
Sentença: ,
Sentença: o
Sentença: que
Sentença: voce
Sentença: pode
Sentença: fazer
Sentença: hoje
2024.1.08.020@suporte-OptiPlex-3050:~$
```

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

```
teste.sh
~/
Open  Save  -  □  ×
1 #!/bin/bash
2
3 i=50
4
5 while [ "$i" -ge 20 ]; do
6     echo "Contagem: $i"
7     i=$((i - 1))
8 done
9
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash teste.sh
Contagem: 50
Contagem: 49
Contagem: 48
Contagem: 47
Contagem: 46
Contagem: 45
Contagem: 44
Contagem: 43
Contagem: 42
Contagem: 41
Contagem: 40
Contagem: 39
Contagem: 38
Contagem: 37
Contagem: 36
Contagem: 35
Contagem: 34
Contagem: 33
Contagem: 32
Contagem: 31
Contagem: 30
Contagem: 29
Contagem: 28
Contagem: 27
Contagem: 26
Contagem: 25
Contagem: 24
Contagem: 23
Contagem: 22
Contagem: 21
Contagem: 20
2024.1.08.020@suporte-OptiPlex-3050:~$
```

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:



```
testenumero.sh
~/
Open ▾  Save  ≡  -  □  ×

1 #!/bin/bash
2
3 i=11
4
5 echo "Verificando um número e sua posição"
6 if [ "$i" -le 20 ] && [ "$i" -ge 10 ]; then
7     echo $((2 * i))
8 elif [ "$i" -ge 0 ] && [ "$i" -le 10 ]; then
9     echo $((3 * i))
10 else
11     echo "Número não está entre 0 e 10 ou entre 10 e 20"
12 fi
13
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ bash testenumero.sh
Verificando um número e sua posição
22
2024.1.08.020@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:

\$# é uma variável especial em shell scripting que representa o número de argumentos passados para um script ou uma função

A screenshot of a code editor window titled 'time.sh' with a dark theme. The editor shows a shell script with 16 lines. Line 1 is the shebang '#!/bin/bash'. Line 3 is an if statement 'if ["\$#" -eq 2]; then'. Lines 5 and 6 assign the first and second arguments to 'verdade1' and 'verdade2' respectively. Line 8 is an echo statement comparing the two variables. Line 9 is an 'else' statement. Line 10 is an echo statement 'Digite dois times:'. Lines 11 and 12 use 'read' to capture user input into 'time1' and 'time2'. Line 14 is an echo statement comparing the two inputs. Line 15 is the 'fi' statement. Line 16 is empty.

```
1 #!/bin/bash
2
3 if [ "$#" -eq 2 ]; then
4
5     verdade1="$1"
6     verdade2="$2"
7
8     echo "Palmeiras é maior do que $verdade1 e $verdade2"
9 else
10    echo "Digite dois times:"
11    read time1
12    read time2
13
14    echo "Palmeiras é maior do que $time1 e $time2"
15 fi
16
```

```
2024.1.08.020@suporte-OptiPlex-3050:~$ gedit time.sh
2024.1.08.020@suporte-OptiPlex-3050:~$ chmod 755 time.sh
2024.1.08.020@suporte-OptiPlex-3050:~$ ./time.sh
Digite dois times:
Flamengo
Corinthians
Palmeiras é maior do que Flamengo e Corinthians
2024.1.08.020@suporte-OptiPlex-3050:~$
```

