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

Nome: Mileno Oliveira Matos **RA: 2021.1.08.051**

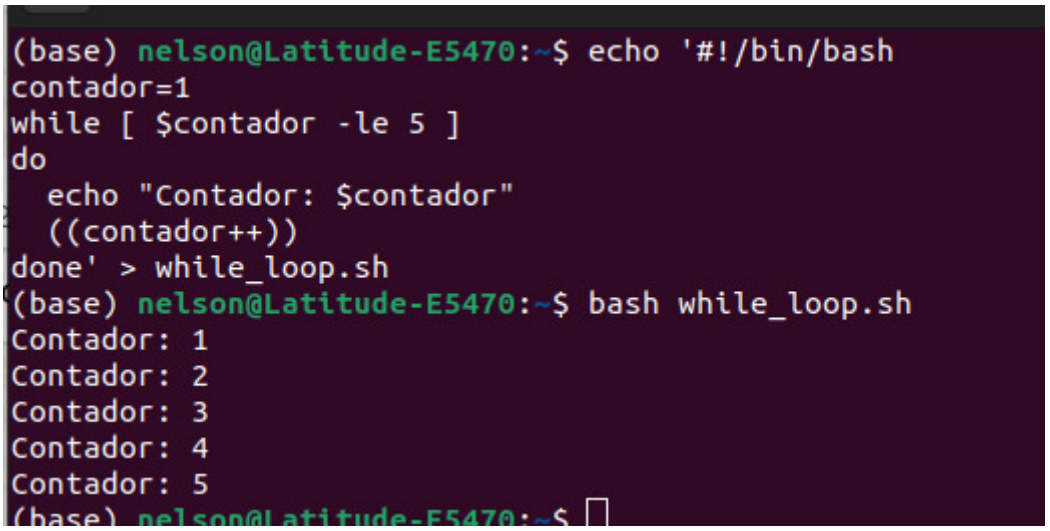
1-

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
echo "Mileno Oliveira Matos"' > escrevenome.sh
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
if [ -r escrevenome.sh ] && [ -x escrevenome.sh ]; then
  echo "Você tem permissão de leitura e execução sobre o escrevenome.sh"
else
  echo "Você NÃO tem permissão de leitura e/ou execução sobre o escrevenome.sh"
fi' > testecompara.sh
(base) nelson@Latitude-E5470:~$ bash escrevenome.sh
Mileno Oliveira Matos
(base) nelson@Latitude-E5470:~$ bash testecompara.sh
Você NÃO tem permissão de leitura e/ou execução sobre o escrevenome.sh
(base) nelson@Latitude-E5470:~$
```

2-

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
fruta=$1
case "$fruta" in
  uva) echo "A uva é o fruto da videira. Usada para vinho, sucos e doces." ;;
  banana) echo "A banana é rica em potássio, ideal para energia rápida." ;;
  maçã) echo "A maçã é uma fruta crocante e saudável, com antioxidantes." ;;
  morango) echo "O morango é pequeno, vermelho e muito usado em sobremesas." ;;
  abacaxi) echo "O abacaxi é tropical, com sabor doce e ácido, ótimo para sucos.
" ;;
  *) echo "Fruta desconhecida." ;;
esac' > frutascase.sh
(base) nelson@Latitude-E5470:~$ bash frutascase.sh morango
O morango é pequeno, vermelho e muito usado em sobremesas.
(base) nelson@Latitude-E5470:~$
```

3-While:O while executa um bloco de comandos enquanto a condição for verdadeira. É ideal quando não sabemos exatamente quantas vezes o laço será executado.



For: O for é usado para repetir um bloco de comandos para cada item de uma lista. Pode ser usado em dois estilos: tradicional (com lista) ou estilo C (com controle por índice).

Until: O until é parecido com o while, mas inverte a lógica: ele executa o bloco até a condição se tornar verdadeira (ou seja, roda enquanto a condição for falsa).

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
x=1
until [ $x -gt 5 ]
do
  echo "x é: $x"
  ((x++))
done' > until_loop.sh
(base) nelson@Latitude-E5470:~$ bash until_loop.sh
x é: 1
x é: 2
x é: 3
x é: 4
x é: 5
(base) nelson@Latitude-E5470:~$
```

4-IFS é a variável que define o separador padrão para leitura de strings. O padrão é espaço, tabulação e newline.

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
texto="pedra,papel,tesoura"
IFS=","
for brincadeira in $texto
do
  echo "Posição: $brincadeira"
done' > ifs_exemplo.sh
(base) nelson@Latitude-E5470:~$ bash ifs_exemplo.sh
Posição: pedra
Posição: papel
Posição: tesoura
(base) nelson@Latitude-E5470:~$
```

5-

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
for ((i=50; i>=20; i--))
do
  echo $i
done' > for_estilo_c.sh
(base) nelson@Latitude-E5470:~$ bash for_estilo_c.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
(base) nelson@Latitude-E5470:~$
```

6-

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
valor=$1
if [ $valor -ge 0 ] && [ $valor -le 10 ]; then
  echo "Triplo: $((valor * 3))"
elif [ $valor -gt 10 ] && [ $valor -le 20 ]; then
  echo "Dobro: $((valor * 2))"
else
  echo "Valor fora do intervalo permitido."
fi' > valor_intervalo.sh
(base) nelson@Latitude-E5470:~$ bash valor_intervalo.sh 8
Triplo: 24
(base) nelson@Latitude-E5470:~$ bash valor_intervalo.sh 15
Dobro: 30
(base) nelson@Latitude-E5470:~$ bash valor_intervalo.sh 25
Valor fora do intervalo permitido.
(base) nelson@Latitude-E5470:~$
```

7-

```
(base) nelson@Latitude-E5470:~$ echo '#!/bin/bash
if [ $# -ne 2 ]; then
  echo "Uso: $0 param1 param2"
else
  echo "Parâmetro 1: $1"
  echo "Parâmetro 2: $2"
  soma=$(( $1 + $2 ))
  echo "Soma dos parâmetros: $soma"
fi' > parametros.sh
(base) nelson@Latitude-E5470:~$ bash parametros.sh 21 4
Parâmetro 1: 21
Parâmetro 2: 4
Soma dos parâmetros: 25
(base) nelson@Latitude-E5470:~$
```

É uma variável especial no Bash (e em outros shells) que representa a quantidade de parâmetros (argumentos) passados para o script na hora da execução. Ela verifica se o número de argumentos está correto, controla a lógica do script dependendo da quantidade de argumentos e evita erros quando faltam parâmetros.