

## Exercícios

Os exercícios abaixo devem ser solucionados com linguagens Shell, como bash, nushell, zsh, fish, etc.

Na décima página, um exercício é proposto – *e erroneamente solucionado* – com o intuito de entender melhor os filtros em Shell scripting:

0. Suponha que exista um arquivo chamado `hotel.txt` com 100 linhas de dados. Imprima começando da linha 20 à linha 30 e armazene este resultado em um novo arquivo `hlist`.

```
#!/usr/bin/env bash
```

NUSHELL

```
$ tail +20 < hotel.txt | head -10 > hlist
```

```
#!/usr/bin/env nu
```

NUSHELL

```
$ open hotel.txt | lines | skip 20 | first 10 | save hlist
```

---

Os exercícios a seguir são da página 44 em diante, e devem ser feitos escrevendo um script shell.

1. Adicione dois números que são recebidos pela linha de comando como argumentos, e se esses dois números não são dados, mostre um erro e seu uso correto:

```
#!/usr/bin/env nu
```

NUSHELL

```
# Add two numbers and return its sum.
```

```
def main [
```

```
  a: int # First number to add.
```

```
  b: int # Second number to add.
```

```
] : nothing -> int {
```

```
  $a + $b
```

```
}
```

2. Retorne o maior número entre três argumentos dados pela linha de comando, e se três argumentos não forem dados, mostre um erro e seu uso correto.

```
#!/usr/bin/env nu
```

NUSHELL

```
# Return the greastest number between the three given.
```

```
def main [
```

```
  a: int # First number to analyse.
```

```
  b: int # Second number to analyse.
```

```
  c: int # Third number to analyse.
```

```
] : nothing -> int {
```

```
  [ $a $b $c ] | sort | last
```

```
}
```

3. Imprima a sequência 5, 4, 3, 2, 1 utilizando a repetição `while`:

```
#!/usr/bin/env bash
```

NUSHELL

```
m=5;
while [ $m -gt 0 ]
do
    echo "$m"
    m=`expr $m - 1`
done
```

```
#!/usr/bin/env nu
```

NUSHELL

```
# Create the sequence 5, 4, 3, 2, 1 using the while loop.
def main []: nothing -> list<int> {
    mut result = []

    mut i = 5
    while $i != 0 {
        $result = $result ++ [ $i ]
        $i = $i - 1
    }

    $result
}
```

3.1. Há outras formas de se resolver sem while, quais seriam?

```
#!/usr/bin/env bash
```

NUSHELL

```
while ((m > 0)); do
    echo "$m"
    m=`expr $m - 1`
done
```

```
#!/usr/bin/env nu
```

NUSHELL

```
# Create the sequence 5, 4, 3, 2, 1 using the command seq.
def main []: nothing -> list<int> {
    seq 5 -1 1
}
```

3.2. E para um máximo e mínimo qualquer dado pela linha de comando, como resolver?

```
#!/usr/bin/env bash
```

NUSHELL

```
if [ $# -lt 2 ]
then
    echo "MAX or MIN missing"
    exit 1
fi
```

```

i=$1

while [ $i -ge $2 ]
do
    echo "$i"
    i=`expr $i - 1`
done

```

```

#!/usr/bin/env nu                                     NUSHELL

# Create a decreasing sequence starting from `max` and ending in `min`.
def main [
    max: int # Start of the sequence.
    min: int # End of the sequence.
]: nothing -> list<int> {
    seq $max -1 $min
}

```

3.3. Imprimir a ordem crescente quando o primeiro argumento for menor que o segundo, senão imprimir a ordem decrescente.

```

#!/usr/bin/env bash                                     NUSHELL

if [ $# -lt 2 ]
then
    echo "MAX or MIN missing"
    exit 1
fi

if [ $2 -gt $1 ]
then
    i=$1
    j=$2

    while [ $i -le $j ]
    do
        echo "$i"
        i=`expr $i + 1`
    done

elif [ $1 = $2 ]
then
    echo "$1"
else
    i=$1
    j=$2

    while [ $i -ge $j ]
    do
        echo "$i"
        i=`expr $i - 1`
    done

```

```
done
fi
```

4. Usando a palavra-chave `case` performe operações matemáticas básicas como adição (+), subtração (-), multiplicação (x), e divisão (/).

```
#!/usr/bin/env nu                                     NUSHELL

# Using case (match) keyword to perform basic math operations such as addition
# (+), subtraction (-), multiplication (x), and division (/).
def main [
  a: int # First operand.
  operator: string # Operator.
  b: int # Second operand.
]: nothing -> int {
  match $operator {
    '+' => { $a + $b },
    '-' => { $a - $b },
    'x' => { $a * $b },
    '/' => { $a / $b },
    _ => { NaN }
  }
}
```

- 4.1. Ao invés de apenas três argumentos, resolva para um número ilimitado de argumentos (podendo ter 1, 2, ... argumentos).

5. Mostre a data, tempo, usuário, e diretório atual.

```
#!/usr/bin/env nu                                     NUSHELL

# Show the current date, time, username, and current directory.
def main []: nothing -> any {
  {
    date: (date now | into string)
    user: (whoami)
    directory: $env.PWD
  }
}
```

6. Faça o reverso do primeiro argumento dado.

```
#!/usr/bin/env nu                                     NUSHELL

# Reverse the given argument.
export def main [
  arg: string # String to be reversed.
]: nothing -> any {
  $arg | str reverse
}
```

7. Dado um número como argumento, calcule a soma de todos os dígitos.

```
#!/usr/bin/env nu                                     NUSHELL

# Given number, calculate the sum of its digits.
export def main [
  arg: int # Number to sum each digit.
]: nothing -> any {
  $arg | into string | split chars | into int | math sum
}
```

8. Performe aritmética real (números com pontos decimais) dado os argumentos.

```
#!/usr/bin/env nu                                     NUSHELL
```