

## Prática 07 – Shell Scripts

---

### Objetivo da Prática:

- Exercitar a prática de Shell Scripts

### Material a ser entregue:

- Para cada script gere um arquivo com o resultado da saída do mesmo e a explicação de qual o seu objetivo. Ao final empacote todos os arquivos.

### I – Condições e Repetição em Scripts:

#### Script 1:

```
#!/bin/bash

nome="joao francisco decio"
for i in $nome
do
    echo $i
    sleep 1
done
```

#### Script 2:

```
#!/bin/bash

for i in `cat arquivo`
do
    echo $i
    sleep 1
done
```

#### Script 3:

```
#!/bin/bash

for i in *
do
    echo $i
    sleep 1
done
```

**Script 4:**

```
#!/bin/bash

for i in $(seq 1 100)
do
    echo $i
    sleep 1
done
```

**Script 5:**

```
#!/bin/bash

for ((i=1; i<=9; i++)); do
    echo -n "$i"
done
```

**Script 6:**

```
#!/bin/bash

inicio=1
fim=10
while [ $inicio -lt $fim ]
do
    echo "Valor inicial é menor que o final: $inicio $fim"
    inicio=$((inicio+1))
done
echo "Valor inicial é igual ao valor final: $inicio $fim"
```

**Script 7:**

```
#!/bin/bash

arq=$1
i=0
echo " ----- Início do arquivo. ----- "
cat $arq | while read LINE
do
    i=$((i+1))
    echo "Linha $i: $LINE"
done
echo " ----- Final do arquivo. ----- "
```

**Script 8:**

```
#!/bin/bash

if [ -f $1 ]; then
    echo "Arquivo encontrado"
else
    echo "Arquivo nao encontrado"
fi

if [ -d $1 ]; then
    echo "Diretorio existe"
else
    echo "Diretorio nao existe ou nao encontrado"
fi
```

### Script 9:

```
#!/bin/bash

echo Informe um nome # tente entender o funcionamento do comando case
read NOME
case $NOME in
    Jose*)
        echo "Boa noite Jose."
        ;;
    Joao*)
        echo "Boa noite Joao."
        ;;
    *)
        echo "Nao te conheco."
        exit 1
        ;;
esac
```

### Script 10:

```
#!/bin/bash

if [ `date +%H` -lt 12 ]
then
    echo Bom dia
elif [ $(date +%H) -lt 18 ]; then
    echo Boa tarde
else
    echo Boa noite
fi
```

### Script 11:

```
#!/bin/bash

HORA=$(date +%H)
if [ $HORA -lt 12 ]
then
    echo Bom dia
elif [ $HORA -lt 18 ]; then
    echo Boa tarde
else
    echo Boa noite
fi
```

### Script 12:

```
#!/bin/bash

echo -n "Deseja limpar a tela: "
read RESPOSTA
if [ ${RESPOSTA} = S -o ${RESPOSTA} = s ]; then
    clear
fi
```

### Script 13:

```
#!/bin/bash

if [ ! -d casa ]; then
    mkdir casa ; cd casa ; touch casa1 casa2 casa3
fi

for original in *; do
    resultado=$(echo $original | tr '[:lower:]' '[:upper:]')
    if [ ! -e $resultado ]; then
        mv $original $resultado
    fi
done
```

## **II – Trabalhando com Strings:**

- a) **length** – obter o tamanho da string ,ou seja, o número de caracteres de uma string; O exemplo seguinte mostra o tamanho de uma string lida. (Digite e teste)

```
#!/bin/bash
echo -n "String: "
read s
echo ${#s}
```

- b) **substring** – extrair parte de uma string;  
O exemplo seguinte lê uma string e mostra parte dos caracteres lidos. (Digite e teste)

```
#!/bin/bash
echo -n "String: "
read s
echo ${s:5}
echo ${s:5:3}
n=${#s}
if [ $(( n % 2 )) -eq 1 ]; then
    m=$(( n / 2 ))
    echo ${s:$m:1}
fi
```

- c) **substituição** – substituir uma parte da string;  
O exemplo seguinte substitui a letra a pela letra x na string lida. (Digite e teste)

```
#!/bin/bash
echo -n "String: "
read s
s1=${s/a/x} ; echo $s1
s1=${s//a/y} ; echo $s1
```

Experimente uma entrada com várias letras a para verificar a diferença entre / e //;

- d) Evidentemente que exercícios como o anterior são mais fáceis de realizar usando os mecanismos da shell que naturalmente separam palavras:

```
#!/bin/bash
echo -n "String: "
read s
p=""
for i in $s ; do
    if [ -z $p ]; then
        p=$i; echo "Primeiro: $p"
    fi
done
echo "Ultimo: $i"
```

### III – Funções em Scripts:

a) As funções são elementos estruturados semelhantes às funções das linguagens de programação clássicas; no caso dos scripts, enquadram um conjunto de comandos que são executados quando a função é invocada através do seu nome; Digite o exemplo a seguir e teste o funcionamento do script:

```
#!/bin/bash
exemplo() {
    echo $FUNCNAME says hello
}

echo "chamar a função..."
exemplo
echo "repete..."
exemplo
```

b) As funções aceitam argumentos numa sintaxe muito semelhante à dos próprios argumentos dos scripts; Digite e teste o script:

```
#!/bin/bash
say () {
    echo "I say, " $1
}

say hello
say hello hello
say "hello hello hello"
```

c) O comando return termina a função, em determinado ponto, permitindo também gerar um valor de retorno; Digite o exemplo a seguir e teste o funcionamento do script:

```
#!/bin/bash
say () {
    if [ $# -eq 0 ]; then
        echo "nothing to say"
        return 1
    fi
    echo "I say, " $*
    return 0
}

say hello
say hello hello
say "helo hello hello"
say
echo just say `say`
```

- d) As variáveis do script estão disponíveis na função; podem ser alteradas tal como podem ser criadas novas variáveis; as variáveis trabalhadas deste modo são todas "globais" (no mesmo sentido usado programação clássica): existem, podem ser criadas e alteradas em qualquer ponto; Digite e teste o script:

```
#!/bin/bash
f () {

    echo "$FUNCNAME : Y= $Y"
    X=77
    Y=88
    echo "$FUNCNAME : X= $X"
    echo "$FUNCNAME : Y= $Y"
}
Y=66
echo "Y= $Y"
f
echo "X= $X"
echo "Y= $Y"
```

- e) No seguinte script é feita uma função readline que lê uma string. Digite o exemplo a seguir e teste o funcionamento do script:

```
#!/bin/bash

readline () {
    echo "readline..."
    msg=""
    if [ $# -gt 0 ]; then
        msg="$*: "
    fi
    str=""
    while [ -z $str ]; do
        echo -n $msg
        read str
    done
    STR=$str
}

readline "Teste"
echo "Lido: " $STR
```

#### IV – Vetores (Arrays):

O shell permite a utilização de variáveis do tipo Array (Vetor). O item a) está apenas exemplificando não precisa testar e salvar.

- a) Os elementos de um Array podem ser definidos usando a sintaxe

variable[indice] - onde indice é um valor inteiro 0,1,2..etc.

Para obter o valor de um elemento de um array utilize a sintaxe \${variable[xx]}.

### Exemplos de Arrays:

```
v[2]=1
v[3]=ola
v[4]=12 #elementos de um array podem não ser consecutivos ou do mesmo tipo
v[7]="ola mundo" #pode deixar espacos no array
echo ${v[2]}
dias=( domingo segunda terca quarta ) #declaração e inicialização de um array
indice=0
echo "Hoje é ${dias[indice]}"
```

### Exemplo de definição de um Array usando substituição de um comando:

```
files=(`ls`) #saída do comando ls passado para um vetor
echo ${files[2]}
echo ${#files[@]} --numero de elementos do array
```

- b) Não é comum o interesse em usar variáveis indexadas, isoladamente, em vez de variáveis comuns. Normalmente o que se pretende é usar um conjunto de posições contíguas para processamentos iterativos (de forma semelhante aos arrays nas linguagens de programação clássicas).

Digite o exemplo e teste o funcionamento do script:

```
#!/bin/bash

i=0
while [ $i -le 5 ]; do
    num[i]=$RANDOM
    echo "Número : $i ${num[i]}"
    i=$(( $i + 1 ))
done
```

A variável \$RANDOM fornece um número aleatório ("diferente") cada vez que é usada; é gerado um número entre 0 e 2<sup>15</sup>.

- c) Um while deste tipo pode ser escrito de maneira mais familiar com a seguinte sintaxe alternativa, mais adequada para ciclos iterativos;  
Exemplo: o seguinte script gera 6 números aleatórios entre 1 e 20: (Digite e teste)

```
#!/bin/bash

for((i=0; i<=5; i++)); do
    num[i]=$(( 1 + 20 * $RANDOM / 2**15 ))
    echo "Número : $i ${num[i]}"
done
```

- d) Exemplo: o seguinte script gera 6 números aleatórios, entre 1 e 20, apresentando-os por ordem. (Digite e teste)

```
#!/bin/bash

for((i=0; i<=5; i++ ));do
    num[i]=$(( 1 + 20 * $RANDOM / 2**15 ))
```

```
done
for((i=0; i <5; i++ )); do
    for((j=0; j<5; j++ )); do
        if [ ${num[i]} -lt ${num[j]} ] ; then
            x=${num[i]}
            num[i]=${num[j]}
            num[j]=$x
        fi
    done
done

for((i=0; i<=5; i++)) ;do
    echo "Numero: $i ${num[i]}"
done
```