

Introdução à Programação com Shell Script

Dr. Marcos Talau (marcos@talau.info)



26 de abril de 2025



Tópicos

- Introdução
- Sintaxe Fundamental de Scripts
- Controle de Fluxo
- Estruturas de Repetição
- Funções



Introdução


Sobre o minicurso

- Prático
- É seguida a *IEEE Std 1003.1-2001, Standard for Information Technology — Portable Operating System Interface (POSIX)*
 - Revisão da *IEEE 1003.2-1992*
 - Norma para criação de aplicações portáteis
 - Não ao *bashism*



Introdução

- Programação para sistemas UNIX/Linux
- Filosofia dos sistemas
- Diversos sistemas
- Editor texto simples
- Predomina modo texto



O que é shell?

O que é shell?

- Shell = Interpretador de Comandos
- Programa que interpreta o que foi digitado e executa outros programas (comandos)
- Alguns shells:
 - GNU Bash
 - Zsh
 - DASH (foco não é na iteratividade), segue o *IEEE Std 1003.1-2001*

Lógica dos Scripts

- Não existe um “main”
- Execução gera um retorno


\$?

executar comandos:

ls

cat

dmesg

- 
- `#!/bin/sh`
 - Arquivo executável
 - Extensão independe
 - Mas por padrão usa-se `.sh`
 - Ver `ex1.sh`
 - Instruções de scripts também podem ser executadas diretamente no prompt do shell



Sintaxe Fundamental de Scripts



Comentários

- Para comentários utiliza-se #

comentário

comentário

de

múltiplas linhas

Mensagem na tela

- Comando echo

1)

```
echo "Olá mundo!"
```

2)

```
echo "Mensagem  
múltiplas linhas"
```

3)

```
echo -n "Qual seu nome: "
```



Obtendo entrada

- Comando read

```
echo -n "Nome: "
```

```
read nome
```

Variáveis

- Declaração
 - Usa-se nome=valor
 - Inteiro
 - total=100
 - String
 - msg=uma/var/tbm/string
 - msg="Uma var string"
 - Qualquer tipo! Pois ela não tem tipo, só guarda os caracteres após o igual
- Uso

\$nome

Prática

- Faça um programa que solicite do usuário: nome, idade, CPF e endereço. Ao final do programa exiba os dados digitados pelo usuário no formato abaixo

```
#####
```

```
Nome: NOME
```

```
Idade: IDADE
```

```
CPF: CPF
```

```
Endereço: END
```

```
#####
```



Variáveis do Shell

```
echo $PATH
```

```
echo $HOME
```

```
echo $USER ou $LOGNAME
```

- Para ver outras:

```
set
```
- Prática: exibir outras três variáveis



Variáveis Especiais


\$?

- Identificação do processo atual

\$\$

- Identificação do último processo executado em background

\$!

- 
- Nome do shell ou do shell script
\$0 (*zero*)
 - Argumentos recebidos
\$1..\$9
 - Número de argumentos
\$#

- Todos os argumentos
\$* agrupados por “”
\$@ separados por “”
- Remover o argumento mais a esquerda
shift


Exportando Variável


- export: Exporta a variável para outros programas executados a partir do shell atual e também a cria localmente

```
export HOJE=sábado
```

```
export
```

```
echo $HOJE
```

- 
- Crie um script que exporte uma variável com seu nome e depois mostre seu conteúdo
 - Execute o script e verifique o funcionamento
 - Agora use o echo para mostrar seu conteúdo no prompt do shell. Funcionou?

- 
- Para a variável exportada pelo script ser exportada para fora dele deve-se executá-lo com o comando
 `. script.sh`
 - Testar o funcionamento

Variável Somente Leitura

- Variável que não pode ser alterada após sua criação

`readonly var=valor`

- Exemplo:

`readonly TMPDIR=/tmp`



Tarefa

- Fazer um programa que receba nome, idade, cpf e endereço via argumentos e exiba para o usuário utilizando a formatação anteriormente usada
- Crie um script para exibir o número total de argumentos recebidos e o seu conteúdo

Recebendo saída Comando

- Recebendo saída de texto...
- Duas formas clássicas:
 `nome_pc=`hostname``
 `nome_pc=$(hostname)`
- Ver ex2.sh
- Tarefa: fazer um script que exiba a data de hoje no formato dd/mm/aaaa



stdout e stderr

ls > /tmp/saida_ls

mesma coisa que

ls 1> /tmp/saida_ls


- Portanto, 1> indica saída padrão



ls prova.txt > /tmp/saida_ls

cat /tmp/saida_ls

- Arquivo vazio...
- Erro saiu no terminal
- Como direcionar o erro para o arquivo?



```
ls prova.txt 2> /tmp/saida_ls  
cat /tmp/saida_ls
```

- 2> indica a saída de erro
- E se eu quiser direcionar a saída padrão e a saída com erro para o arquivo?



ls arquivo.txt > /tmp/saida_ls 2>&1

Cálculos matemáticos simples

`$(n1 OP n2)`

- Operações

`+`

`-`

`/`

`*`

`%` (resto da divisão)

- Testar:

`echo $(2 + 2)`

`echo $(3 * 2)`

`echo $(5 % 2)`

Cálculos matemáticos simples com expr

expr n1 OP n2

- Operações

+

-

/

* (usar \)

% (resto da divisão)

- 
- Tarefa: fazer um script que realize a soma de dois números via entrada por argumentos



Controle de Fluxo



|| e &&

```
cd /tmp
```

```
touch arq_novo
```

```
rm arq_novo
```

```
echo $?
```

```
rm arq_novo
```

```
echo $?
```

- Lembrando, \$? retorna o status do comando. 0 = sucesso.



Uso do ||

- O || só executa o próximo comando em caso de falha

```
rm arq_novo || echo "Falha ao remover"
```



Uso do &&

- Só executa o próximo comando em caso de sucesso

```
test -f /usr/bin/wget && wget www.google.com
```



Condicional if

Condicional if

- Sintaxe cmd:

```
if cmd; then
```

```
...
```

```
elif cmd; then
```

```
...
```

```
else
```

```
...
```

```
fi
```

- Ver ex3.sh

- 
- Sintaxe variáveis:

```
if [ "a" = "a" ]; then
```


```
    echo "iguais"
```

```
fi
```

= igual

!= diferente

- Sintaxe pode ser vista em `man test`
- Ver ex4.sh

- 
- Tarefa 1: Fazer um script que obtenha a hora e os minutos do usuário e diga se ela é válida. Considere as horas no intervalo 00-23
 - Tarefa 2: Fazer um script que verifique a existência do diretório Downloads no home do usuário



Case

case

```
case $var in
    "op1")
        cmd
        ;;
    "op2")
        cmd2
        ;;
    *)
        echo "outra op"
        ;;
esac
```

- Ver ex5.sh
- Tarefa: Fazer script que receba uma operação matemática (+ - / *) com dois números e retorne o resultado.



pipe |

Pipe

- Joga a saída de um comando para a entrada de outro
- Exemplos:

```
cat /etc/passwd | grep `whoami`
```

```
ls | sort
```

```
ls . | wc -l | { read x; echo "Existem $x arquivos"; }
```

- Sobre o { cmd; }
 - Agrupa comandos. Exemplo de utilidade:
false && echo oi; echo oi2
false && { echo oi; echo oi2; }
true && { echo oi; echo oi2; }



Estrutura de Repetição for

For

```
for var in lista; do  
    echo $var  
done
```

- Ver ex6.sh
- Tarefa: Faça um script que exiba os números ímpares de 1 a 50



while

while

- while = Enquanto for, faça
- while cond; do
 ...
done
- Ver ex7.sh



until

until

- until = Faça até que
until cond; do

...

done

- Ex:

c=1

until [\$c = 10]; do

 echo "var c é diferente de 10"

 c=10

done



Exercício

- Faça um script que para uma lista de números qualquer diga quais números são menores que n .



Funções



Funções

- Sintaxe
- Parâmetros
- Alteração de variáveis
- Retornos
- Bibliotecas

Sintaxe

```
nome_funcao()  
{  
    # conteúdo da função  
}
```

- Para chamar:
 nome_funcao
- Testar função simples



Parâmetros

```
exibe_nome()  
{  
    echo "Olá $1"  
}
```

```
exibe_nome Linux
```

Alteração de variáveis

```
altera_x()  
{  
    echo "x era igual a $x"  
    x=90  
}
```

```
x=5  
altera_x  
echo "Agora x = $x"
```

Variáveis locais

```
altera_x()  
{  
    local x  
    echo "x é igual a $x"  
    x=90  
    echo "x é igual a $x"  
}
```

```
x=5  
altera_x  
echo "Agora x = $x"
```


Retorno

```
soma()  
{  
    echo -n "$1 + $2 = "  
    return `expr $1 + $2`  
}
```

```
soma 2 2; echo $?
```

```
soma 20 5; echo $?
```

Bibliotecas

```
#### arquivo lib.sh ####  
checa_root()  
{  
    test `whoami` = root || exit 1  
}
```

```
#### arquivo meuScript.sh ####  
. ./lib.sh
```

```
checa_root  
ls -a /root
```



Exercício

- Crie uma biblioteca que contenha as funções:
 - `checa_root`
 - `lista_arquivos`
 - `lista_diretórios`
 - `exibe_argumentos`

Introdução à Programação com Shell Script

- Slides disponíveis em:



http://talau.info/public_talks