Shell Script para iniciantes

whoami

- Caio Jordão Carvalho
- Análise e Desenvolvimento de Sistemas – IFBA
- Labrasoft
- Github: cjlcarvalho

whatis shell

- É o "prompt" da linha de comando do Unix e Linux, ou seja, recebe os comandos digitados pelo usuário e os executa.
- Apresenta recursos e características de linguagem de alto nível, permitindo a criação de scripts para automatização de tarefas e rotinas.

cat scripts.txt

- Automatização de uma série de tarefas que podem ser realizadas por um humano uma por vez.
- Geralmente são executados através de linguagens interpretadas.
- Rápidas para aprender e fáceis de escrever.

hello world

echo “Olá, mundo!”

variáveis

```
variavel=”hoje é dia de flisol!”
echo $variavel
idade=” bem vindos!”
echo $variavel$idade
dia=8
echo “$dia de abril”
```

variáveis

```
hoje=$(date)
echo "Data: $hoje"
unset hoje
echo $hoje
num=2
soma=$(($num + $num))
echo "$num + $num é $soma"
```

arrays

```
salada=( tomate cebola alface )
echo $salada(1)
echo $salada(2)
echo $salada(3)

contagem=( 1 2 3 4 5 6 )
```

condicionais

- Testes em variáveis:

  Números:

  -lt → (9 -lt 10)

  -gt → (10 -gt 9)

  -le → (10 -le 11)

  -ge → (11 -ge 10)

  -eq → (10 -eq 10)

  -ne → (11 -ne 10)

condicionais

Strings:

= → (“casa” = “casa”)

!= → (“amarelo” != “azul”)

-n → (-n “amarelo”)

-z → (-z “”)

condicionais

- Testes em arquivos:

  -d → (-d “/user/home”)

  -f → (-f “/user/home/arquivo.txt”)

  -r → (-r “/user/home/legivel.txt”)

  -s → (-s “/user/home/nao-vazio.txt”)

  -w → (-w “/user/home/pode-escrever.txt”)

condicionais

-nt → (“novo.txt” -nt “velho.txt”)

-of → (“velho.txt” -ot “novo.txt”)

-ef → (“novo.txt” -ef “novo.txt”)

condicionais

- E (AND) lógico:

  -a
- Ou (OR) lógico:

  -o

if-else

```
if ( condicao ); then
    # comandos
else
   # comandos
fi
```

switch case

```
echo “Entre com um número: “
read num
case $num in
    1) echo “um” ;;
    2) echo “dois” ;;
    3) echo “três” ;;
    *) echo “...” ;;
esac
```

while

```
while ( condicao ); do
    # comandos
done
```

until

- Continua executando o loop enquanto determinada condição não for verdadeira.

```
until ( condicao ); do
    # comandos
done
```

for

```
for (( i=1; $i < 10; i=$(($i + 1)) )); do
    # comandos
done
```

for

```
arr=( 1 2 3 4 5 )
for i in "${arr(@)}"; do
    echo $i
done
for i in {1..10}; do
   echo $i
done
```

for

```
for i in $(ls); do
    echo $i
done
```

funções

```
funcao () {
    echo “oi”
}

funcao
```

funções

```
funcao () {
    echo $1
    echo $2
    echo $3
    if ( $# -gt 3 ); then
        echo “chega”
    fi
}
funcao 1 2 3 4 5 6
```

funções

```
if ( $1 = “python” ); then
    echo “py”
else
    echo “sh”
fi
```

funções

```
funcao () {
    return 1
}
funcao
var=$?
echo $var
```

funções

ifconfig

echo $?

mkdir “pasta”

echo $?

sudo apt-get update

echo $?

cat bash-commands.txt

- O bash permite executar desde grandes aplicações até pequenas rotinas gerenciadoras do sistema operacional.
- É importante para o sysadmin ou programador sempre entender comandos do sistema operacional que podem facilitar ou auxiliar essas tarefas.

echo

- O comando "echo" imprime textos na tela. Similar ao "print" de outras linguagens de programação.
- Exemplo:

  echo "Instalação completa"

  echo "Bem vindo ao Flisol"

  echo "Olá"

echo

- É possível declarar variáveis no shell e mandar o echo imprimí-las.
- Exemplo:

  a=”Hoje é dia de Flisol!”

  echo $a

  b=5

  echo “Imprimindo o número $b”

echo

- Além disso, o echo pode ser utilizado para escrever em arquivos.

  echo “Criando um arquivo novo” > new.txt

- **Lembre-se**: > é diferente de >>

  echo “Adicionando uma nova linha” >> old.txt

read

- Comando para a entrada de dados. Você pode utilizá-lo para atribuir valores a variáveis.

```
echo “Digite a sua idade: “
read idade
echo “Você tem $idade anos.”
```

cd

- Comando utilizado para acessar diretórios.

```
cd folder
cd /home/user/Downloads/pasta
cd /
cd ~/
cd ..
```

ls

- Lista o conteúdo do diretório.

ls

ls -las

ls -R

mv

- Comando utilizado para mover ou renomear arquivos ou diretórios do sistema.

mv arquivo.txt arquivo-bkp.txt

mv pasta pasta-bkp

mkdir

- Comando para criar novos diretórios.

  mkdir pasta

- Para criar uma pasta dentro de outra, basta passar o parâmetro -p.

  mkdir -p pasta-pai/pasta-filha

cp

- Utilizado para copiar arquivos e diretórios.

  cp arquivo.txt /pasta/novo-local/novo.txt

- Utilize -r para copiar diretórios.

  cp -r /diretorio /home/user/novo-local

rm

- Utilizado para remover arquivos e diretórios.

rm arquivo.txt

rm -rf /dir

cat

- Utilizado para mostrar o texto presente em arquivos.

  cat meu-arquivo.txt

head

- Comando utilizado para ver apenas as primeiras dez linhas de um arquivo.

  head texto.txt

tail

- Utilizado para ver apenas as últimas dez linhas de um arquivo.

  tail log.txt

- **Lembrete:** Muito útil para acompanhar logs de aplicações.

more / less

- Fazem com que você possa percorrer pelo último comando.

- **Observação:** Muito útil em servidores onde a interface é via terminal.

```
cat loginfo.txt | more
cat loginfo.txt | less
```

chmod

- Altera permissões de acesso a arquivos e diretórios.

- Níveis de permissão:

  Permissão: rwx (leitura-escrita-execução)

  Em binário: 111

  Octal: 7

chmod

Permissão: r-x (leitura-execução)

Em binário: 101

Octal: 5

Permissão: --x (execução)

Em binário: 001

Octal: 1

chmod

- Lembrando: o chmod fornece permissão para o usuário, grupo e outros. Então caso você utilizar:

  chmod 751 arquivo.txt

- Estará dando permissão completa para o usuário, de leitura-execução para o grupo e apenas de execução para outros.

chown

- Altera o proprietário e o grupo de arquivos e diretórios.

chown user diretorio

chown user arquivo

grep

- Recebe uma string e pode ser concatenado com um outro comando para retornar linhas que possuam palavras semelhantes à string passada.

```
ls | grep 'arquivo'
cat texto-gigante.txt | grep 'info'
lspci | grep 'VGA'
```

man

- Manual de referência para comandos e aplicações. Irá retornar toda a documentação do mesmo.

  man ifconfig

- Observação: “man 3” retorna a documentação de funções do C/C++. Exemplo:

  man 3 malloc

VIM (vi improved)

FIM