



# Shell Script



# Conteúdo Programático da aula

- Shell Script (Utilizar nano para criação dos scripts)
- echo “Mensagem”
- printf “Mensagem \n”
- sleep
- read
- Conceito de variáveis no shell script (variáveis de ambiente e criadas por usuário)
- Diferença entre: aspas, apóstrofe e crase.
- Realizando cálculos com \$(( ))

Shell Script é uma linguagem de programação muito utilizada no Linux e interpretada pelo shell. O shell por sua vez é o interpretador de comandos do Linux, ele é responsável pela interpretação de todos os comandos digitados no modo texto.

A linguagem suporta desde comandos simples até poderosos scripts de configuração para servidores.

Mas não podemos deixar de mencionar que o Shell Script também pode ser compilado através de compiladores como: o shc ou c-shell. Um compilador vai transformar um arquivo com a extensão “.sh” em arquivo binário.

Para criar um Shell Script podemos simplesmente



digitar comandos no shell ou utilizar um editor de texto em modo gráfico ou em modo texto, nele vamos inserir os comandos que o script deverá executar. Imagine que Shell Script nada mais é que um arquivo de texto executável.

Através de umscript conseguimos automatizar tarefas no sistema operacional. Isso facilita a vida de um administrador de redes ou de servidores, por exemplo.

Uma característica importante do Shell Script é que podemos utilizar as seguintes estruturas:

- ❖ Estruturas de decisão;
- ❖ Estruturas de repetição;



- ❖ Funções;
- ❖ Variáveis;

As estruturas de decisão (if, switch) e estruturas de repetição (for, while) são comuns na maioria das linguagens de programação. Para que o aprendizado de Shell Script seja mais rápido, utilizaremos o editor de texto nano, devido a sua facilidade.



# INICIANDO NO SHELL SCRIPT

Escolas e Faculdades



Todo programa deve começar com um interpretador de script. O programa inicia da seguinte maneira:

```
#!/bin/bash
```

Ou

```
#!/bin/sh
```

*#: O sustenido, quando usado no início de uma linha é entendido como comentário pelo Shell.*

/bin/sh : Esse é o local do Shell no sistema. Esse comentário “#!/bin/sh” vai mostrar para o Shell que o arquivo é um script e que deve ser executado no shell.

Após o comentário mostraremos para o Shell o que ele deve fazer. Por convenção os arquivos que serão scripts deverão ser salvos com a extensão “.sh”.

Para executar um script através do modo gráfico, podemos simplesmente dar dois cliques para abri-lo. Automaticamente será aberto o terminal e o mesmo será executado. No modo texto podemos executar de duas formas, são elas:

```
sh nome_do_script.sh
```

ou

```
./nome_do_script.sh
```



Todo arquivo de script deve ser executável, ou

Escolas e Faculdades



seja, ele deve ter a permissão de execução pelo menos para o usuário (dono). Para isso pode ser utilizado o comando `chmod`.

NoShell Script utilizamos o comando “sleep”, responsável por fazer uma pausa no decorrer do script. O tempo para essa pausa pode ser informado através de segundos.

Mas para um script ficar completo, podemos ainda enviar informações para o usuário ou solicitar dados vindos do mesmo. Mas como fazer isso?

# Exibindo valores

Para enviar mensagens para o usuário, utilizamos os comandos “echo” ou “printf”. Podemos imprimir uma variável do sistema através destes comandos.

Podemos utilizar os códigos:

\n – para indicar uma quebra de linha no terminal

\t – para indicar uma tabulação no terminal

*Exemplo de uso do echo*

```
echo "Oi usuário"
```

*Exemplo de uso do printf*

```
printf "Oi usuário \n"
```



```
#!/bin/bash
clear
echo "*****"
echo "*"
echo "*** Oi, como vocês estão? Tudo bem com vocês? ***"
echo "*"
echo "*****"
ls
history
free -g
echo
echo
echo
echo "Fim da execução so script do Linux"
```



# Recebendo valores

Para receber informações digitadas pelos usuários, utilizamos o comando “read”. Ao utilizar o comando “read” devemos armazenar o valor recebido em algum lugar, no caso armazenaremos o valor em variáveis.

# Variáveis de ambiente x variáveis do usuário

No Linux temos dois tipos de variáveis, as variáveis de ambiente gerenciadas pelo sistema e as variáveis de usuário, aquelas que nós mesmos declaramos.

As **variáveis** são endereços de memória alocados temporariamente na memória RAM do computador (onde ficam armazenados valores temporários). As variáveis no Linux são precedidas pelo caractere “\$” (cifrão ou dollar como preferir).

As variáveis do sistema são criadas em letras maiúsculas. Para visualizar as variáveis de ambiente, utilizamos os seguintes comandos:

```
$printenv
```

ou

```
$env
```

# Exemplos de variáveis do sistema

**SHELL:** Exibe o nome do shell

**PWD:** Exibe o diretório corrente

**HOME:** Exibe o diretório home do usuário

**LOGNAME:** Exibe o login do usuário



# Criando uma variável

Para criarmos uma variável, podemos utilizar qualquer letra do alfabeto, lembrando que o shell é case sensitive, ou seja, ele diferencia letras maiúsculas de minúsculas, caracteres especiais, etc.

*Não é permitido iniciar o nome da variável com números.*

Para exibir um valor de uma variável precisamos colocar o sinal de cifrão (dólar) \$ antes do seu nome.

Para criarmos uma variável basta digitar o nome da variável e atribuir um valor através do sinal de “=” igual.

*Exemplo: nota1 recebe 10, ou seja, o valor 10 será atribuído à variável nota1.*

```
nota1=10
```

O valor pode ser expresso entre as aspas (“”), apóstrofes (‘) ou crases (`).

Terminal

andrea@andrea-VirtualBox: ~

GNU nano 2.2.6 Arquivo: var.sh

```
#!/bin/bash
clear
variavel1=100
variavel2=200
variavel3=300
variavel4=$(( $variavel1 * ($variavel2 * $variavel3) ))
echo "o valor da variavel final é :$variavel4"
```

[ 7 linhas lidas ]

^G Obter A ^O Gravar ^R Ler o A ^Y Pág Ant ^K Recort ^C Pos Atual  
^X Sair ^J Justifi ^W Onde es ^V Próx Pág ^U Colar T ^T Para Spell

Mas qual é a diferença?

- Aspas interpretam os valores que estiverem dentro das variáveis;
- Apóstrofes lerão o valor literalmente, sem interpretar nada;
- As crases vão interpretar um comando e retornar a sua saída para a variável.



*Obs: Para realizar cálculos no shell utilizamos \$((insira o cálculo aqui))*

*ATENÇÃO: O caracter especial “\” (contra barra) serve para “escapar” a interpretação, ou seja, não vai ser impresso o valor da variável, e sim o nome da mesma.*

*Exemplo: realizando um cálculo*

```
thiagocury@tcury-note:~$ echo $((5+6))
```

*Exemplo: realizando um cálculo*

```
thiagocury@tcury-note:~$ echo $((5+6))
```

*Exemplo: sequência de comandos, criando variáveis com valores e calculando com as variáveis*

```
thiagocury@tcury-note:~$ valor1=5
thiagocury@tcury-note:~$ valor2=6
thiagocury@tcury-note:~$ total=$((valor1+valor2))
thiagocury@tcury-note:~$ echo $total
11
thiagocury@tcury-note:~$
```

*Exemplo: exibindo variáveis de sistema juntamente com um texto*

```
echo "Você está neste momento no diretório: $PWD"
```

**Cuidado** ao utilizar **aspas**(conhecida popularmente por **aspas duplas**) ou **apóstrofe**(conhecida por **aspas simples**).

Escolas e Faculdades



Observe a seguir exemplos de exibição de mensagens utilizando aspas duplas e aspas simples.

```
echo "caminho atual: $PWD"
```

Saída no terminal: **caminho atual: /home/thiagocury**

```
thiagocury@tcury-note:~$ echo 'caminho atual: $PWD'
```

Saída no terminal: **caminho atual: \$PWD**

```
thiagocury@tcury-note:~$ echo caminho atual: $PWD
```

Saída no terminal: **caminho atual: /home/thiagocury**

```
thiagocury@tcury-note:~$ echo "utilizando variável: \ $PWD vou  
mostrar o caminho atual: $PWD"
```

Saída no terminal: **utilizando variável: \$PWD vou mostrar o caminho  
atual: /home/thiagocury**



## Exemplos de scripts

---

### **Exemplo 1:**

*Script mostrando o calendário e a data do sistema.*

```
#!/bin/bash  
clear  
date  
sleep 5s  
clear  
cal  
sleep 5s  
clear
```



## **Exemplo 2:**

*Script avisando o usuário que está mostrando o calendário e a data do sistema. Esse script envia mensagens através do comando echo e printf.*

```
#!/bin/bash
clear
printf "Mostrando a data\n"
date
sleep 5s
clear
echo "Mostrando o calendário\n"
cal
sleep 5s
clear
```



### **Exemplo 3:**

*Script que solicita o nome do usuário e após, armazena em uma variável chamada "nome" e mostra o conteúdo da variável para o usuário.*

```
#!/bin/bash  
clear  
printf "digite seu nome: "  
read nome  
printf "seu nome é: $nome"
```