




[www.4LINUX.com.br](http://www.4LINUX.com.br)

**Só na 4Linux você  
aprende  
MUITO MAIS!**

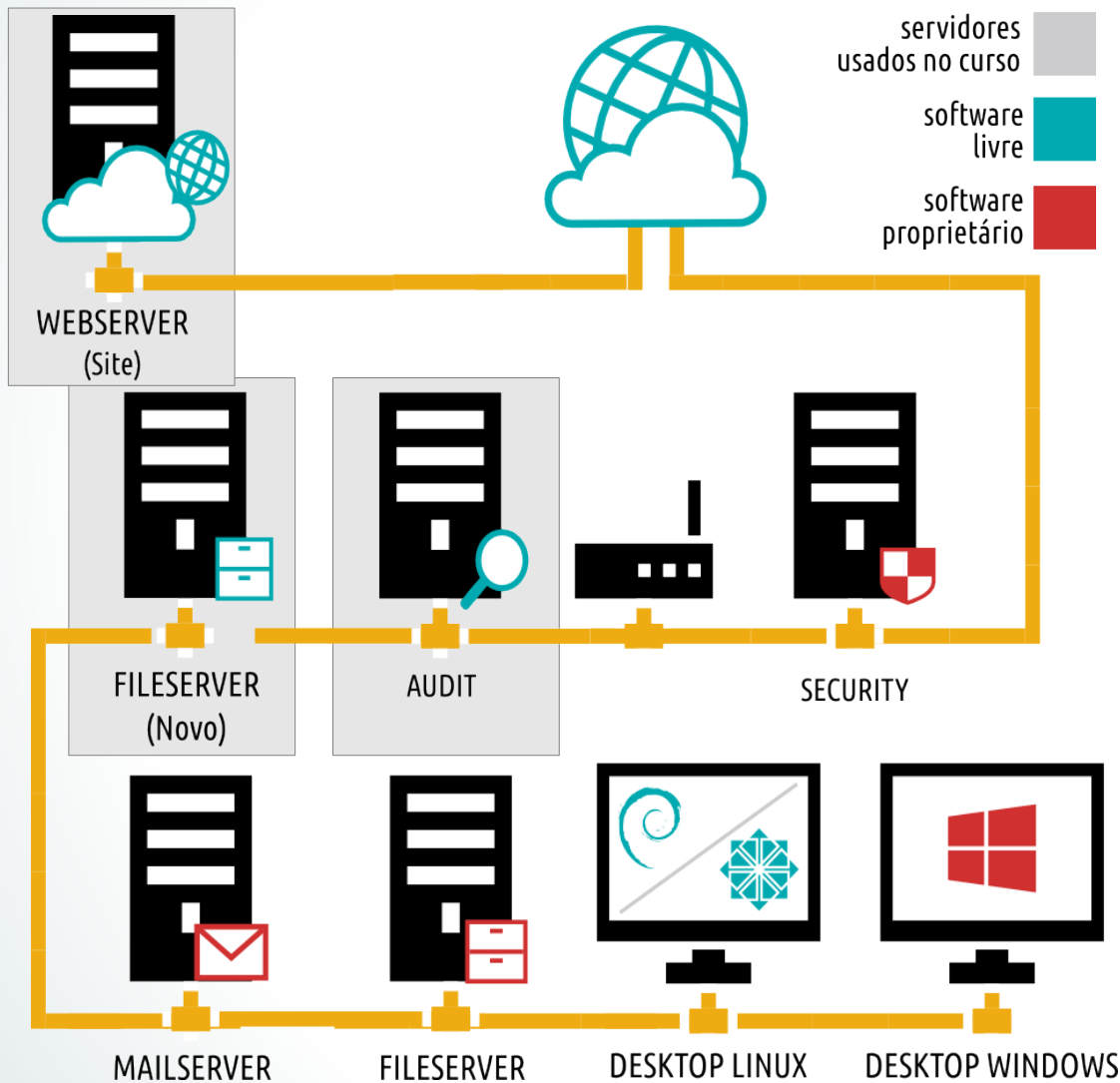
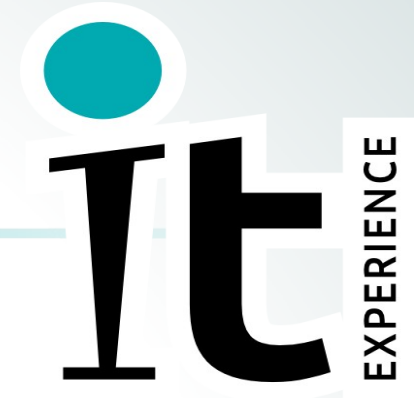
# Shell Script

---

A graphic of a computer monitor with a silver frame and a dark screen. The screen displays a shell script in white text.

```
#!/bin/bash  
echo "Hello World"
```

# IT Experience



## Nesta Aula:

- WebServer (Intranet) – Local  
Acesso pelo VirtualBox  
SO: CentOS Linux



# Objetivos da Aula

---

- Introdução ao Shell Script;
- Conhecer variáveis especiais e operadores;
- Criar scripts utilizando as principais estruturas do shell (if;case,while e for).
- Criar estruturas de backup utilizando shell script;

# Shell Script

- Shell Script são arquivos textos que possuem uma sequência de comandos para atender algum tipo de automatização.
- Por nomenclatura (Não é obrigatório) um shell script recebe a extensão .sh;
- Existem 3 regras básicas na estrutura de um Shell Script:
  - 1 → Primeira Linha referenciar o Shell que irá interpretar o Shell.
  - 2 → Permissão de execução no arquivo
  - 3 → Por nomenclatura (não é obrigatório) todo shell recebe a extensão .sh

# Shell Script

## ➤ Estrutura do Shell Script

```
1# mkdir /root/shell
```

```
2# cd /root/shell
```

## ➤ Criando nosso primeiro script:

```
3# vim shell.sh
```

```
#!/bin/bash
#Primeiro Shell
clear
cal
date
```

```
4# file shell.sh
```

```
5# chmod +x shell.sh
```

```
6# ./shell.sh
```

O comando **file** informa o tipo de arquivo e sua localização no sistema, scripts serão identificados por este comando como “text executable”.

# Shell Script

## ➤ Uso de variáveis

➤ Exemplo simples manipulando variáveis string e números:

```
1# vim script1.sh
```

```
#!/bin/bash
USUARIO=aluno
echo $USUARIO
A=10
B=20
echo "O resultado da soma é `expr $A + $B`"
echo "O resultado da soma é $(expr $A + $B)"
echo 'O resultado da soma é $(expr $A + $B)'
```



# Shell Script

## ➤ Uso de variáveis

- Exemplo prático com base em nas operações de backup da dexter:

```
1# vim script2.sh
```

```
#!/bin/bash
LISTA=$(awk -F: '$3 >= 500 {print $1}' /etc/passwd)
TODAY=$(date +%d-%m-%Y)

## Exibe usuários com UID maior ou igual a 500
echo $LISTA

## Gera um backup com a data atual
tar cjf /backup/backup-home-$TODAY.tar.bz2 /home
```

# Shell Script

## Variáveis especiais

**\$1** → Armazena primeiro parâmetro, **\$2** → Armazena segundo parâmetro, **\$3** .....

**Sintaxe:** comando <parâmetro> <parâmetro> <parâmetro>

### Exemplo de uso em linha de comando

```
1# echo Curso Linux 4Linux
```

### Exemplo de uso em Shell Script

```
2# vim script3.sh
```

```
#!/bin/bash  
  
echo $1 $2 $3
```

```
3# ./script3.sh Curso Linux 4Linux
```

# Shell Script

## Variáveis especiais

**\$\*** → Armazena todos os parâmetros passados sem ter limites;

**\$#** → Armazena a quantidade de parâmetros passados;

**\$0** → Armazena o nome do script executado.

Exemplo com variáveis especiais:

```
1# vim script4.sh
```

```
#!/bin/bash  
echo "Os parâmetros passados são $*"   
echo "A quantidade de parâmetros passados é $#"  
echo "O nome do script é $0"
```

```
2# ./script4.sh Curso 451 da 4Linux
```

# Shell Script

## Variável \$?

➤ A variável interrogação é conhecida por testar o valor de retorno de qualquer comando quando mostrada após sua execução. Exemplos:

```
1# ls /etc/
```

```
2# echo $?
```

```
3# 4linux
```

```
4# echo $?
```

Resultado é igual a “0” → Comando executado com **sucesso!**;

Resultado **diferente** de “0” → Existiu algum **problema** na execução do comando.

# Shell Script

## ➤ Comando test

- O comando test é usado para realizar testes de condicionais (strings, matemáticas e em arquivos) em “Shell Script”.

### Exemplo com strings:

```
1# test "curso" = "curso" ; echo $?
```

### Exemplo com variável:

```
2# test -z $CURSO ; echo $?
```

### Exemplo com expressões matemáticas:

```
3# test 7 -eq 7 ; echo $?
```

# Shell Script

## Operadores matemáticos:

num1 **-eq** num2 : num1 é igual a num2 (**equal to**)

num1 **-ne** num2 : num1 não é igual a num2 (**not equal to**)

num1 **-gt** num2 : num1 é maior que num2 (**greater than**)

num1 **-ge** num2 : num1 é maior ou igual a num2 (**greater or equal**)

num1 **-lt** num2 : num1 é menor que num2 (**less than**)

num1 **-le** num2 : num1 é menor ou igual a num2 (**less or equal**)

## Operadores para arquivos:

**-e** → O arquivo existe (exists);

**-nt** → O arquivo é mais novo que (newer than);

**-ot** → O arquivo é mais antigo que (older than);

**-d** → O diretório existe.

# Shell Script

## ➤ If

- Com o if é possível utilizar operações lógicas, permitindo que o script tome decisões, ou seja, que excessões sejam tratadas, utilizamos estrutura if pois em determinado momento um script pode apresentar várias possibilidades de saída.

Sintaxe:

```
If [ condição ]; then  
    comandos  
else  
    comandos  
fi
```

A execução dos comandos de uma condição **if** é dividida entre os valores que vem após o **then** ou seja, caso a operação imposta seja verdadeira e os valores colocados após o **else** caso a operação imposta seja dada como falsa (invalidada)

Uma estrutura **if** sempre será finalizada por **fi**.

# Shell Script

## ➤ Uso de estrutura if

➤ Exemplo prático com base nas operações de backup da dexter:

```
1# vim script5.sh
```

```
#!/bin/bash
echo "Digite o nome do usuário para consulta:"
read USUARIO
RESULTADO=$( getent passwd | grep $USUARIO)
test -z $RESULTADO
if [ $? -eq 0 ] ;then
    echo "O usuário $USUARIO não existe!"
else
    echo "O usuário existe"
fi
```



# Shell Script

## ➤ Case

- O "case" é utilizado para comandos de fluxo, tal como é o if, mas enquanto if testa expressões não exatas, o "case" vai agir de acordo com resultados exatos.

Sintaxe:      case <valor> in

    <padrão1>)

    Comandos

    ;;

    <padrão2>)

    Comandos

    ;;

    \*)

    Comandos

    ;;

esac

Toda declaração case deverá terminar com um **esac**.

Os padrões são seguidos de seus respectivos comandos, o término de um bloco de comandos é definido por “;;”

# Shell Script

```
# vim script6.sh
```

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

```
echo -e "
```

```
    1- Item 1
```

```
    2- Item 2
```

```
    3- Item 3 "
```

```
echo -n "Escolha uma opção: "
```

```
read OPT
```

```
case $OPT in
```

```
    1) echo "Executando Funcionalidade 1"
```

```
;;
```

```
    2) echo "Executando Funcionalidade 2"
```

```
;;
```

```
    3) echo "Executando Funcionalidade 3"
```

```
;;
```

```
    *) echo "Opção Invalida"
```

```
;;
```

```
esac
```

# Shell Script

## ➤ While

- O "while" é utilizado para testar continuamente uma expressão, até que ela se torne falsa.

Sintaxe:

```
while [< expressão > ];
```

```
do;
```

```
comandos;
```

```
done
```

# Shell Script

```
1# vim script7.sh
```

```
#!/bin/bash

i=1
While [ $i -le 10 ]; do

Clear
echo $i
Sleep 1
i=$((i+1))
done
```

Declarações do tipo while geralmente são utilizadas para executar uma série de comandos enquanto uma expressão permanecer verdadeira.

Neste exemplo a condição utilizada foi testar se o valor da variável i é menor ou igual a 10 (**-le**).

# Linguagem SQL



# Objetivos da Aula

---

- Introdução ao SGBD;
- Criar banco de dados e tabelas;
- Cadastrar, pesquisar e remover dados;

# Gerenciando de dados SQL

---

## Introdução ao SGBD

- Um SGBD é um sistema de gestor de base de dados, onde disponibiliza uma interface para que clientes (usuários), possam interagir com o banco de dados de varias maneiras como inserir dados, pesquisar, excluir, entre outras.
- Acesse o Banco de Dados MySQL do WebServerInterno:

```
1# mysql -u root -p      (Senha 123456)
2> show databases;
3> CREATE database backup;
4> show databases;
```

# Gerenciando de dados SQL

➤ Acesse a Base backup recém criada:

1> use backup;

2> show tables;

3> <CTRL+L> (Limpa a Tela)

➤ Crie uma tabela que irá armazenar os dados do Backup:

```
4> CREATE TABLE log (  
    id INT NOT NULL AUTO_INCREMENT,  
    inicio TIMESTAMP,  
    fim TIMESTAMP,  
    server VARCHAR(100),  
    arquivo VARCHAR(100),  
    status VARCHAR(5),  
    PRIMARY KEY (id));
```



# Gerenciando de dados SQL

---

## ➤ Veja a tabela recém criada:

1> show tables;

2> desc log;

## ➤ Criando Usuário e Liberando Acesso a Base:

3> GRANT ALL PRIVILEGES ON log.\* TO suporte@localhost  
IDENTIFIED BY '4linux' WITH GRANT OPTION;

4> FLUSH PRIVILEGES;

5> SELECT host,user,password FROM mysql.user;

# Gerenciando de dados SQL

## ➤ Inserindo dados na tabela log:

```
1> INSERT INTO log (inicio,fim,server,arquivo,status) VALUES  
( '2011-06-09 15:21:21', '2011-06-09 15:25:21', 'intranet',  
'etc-2011-06-09-15:21:21.tar.gz', 'OK');
```

```
2> select * from log;
```

**Repita o INSERT ACIMA ALTERANDO O STATUS PRA FAIL**

```
3> select * FROM log;
```

```
4> select server,inicio FROM log where status='OK';
```

# Gerenciando de dados SQL

## ➤ Alterar o nome do Servidor intranet para WebServerInterno:

```
1> UPDATE log SET server="WebServerInterno";
```

```
2> select * from log;
```

## ➤ Apagar o registro com FAIL da base:

```
1> DELETE FROM log WHERE status='FAIL';
```

```
2> select * from log;
```

# Laboratório Dexter



- Iremos agora criar um Shell Script no Servidor WebServerInterno que irá gerenciar o Backup dos Servidores WebServerCloud e WebServerInterno;
- Esse Shell Script deverá ser colocado no Cron para que seja criada uma rotina de Backup Diário;
- Acesse no Browser o Sistema de Backup Web:

**[http://192.168.20\\*.X/bkpreport](http://192.168.20*.X/bkpreport)**

# Laboratório Dexter



➤ Vamos criar o Shell Script que fará backup do Servidor WebServerInterno e agendá-lo no cron.

```
1# cd /etc
2# mkdir backup
3# cd backup
4# vim backupdexter.sh
```

**Acompanhe com o Instrutor**

```
5# vim /etc/crontab

00 4 * * * root /etc/backup/backupdexter.sh /home
10 4 * * * root /etc/backup/backupdexter.sh /etc
```

# Pergunta LPI

---



Qual a palavra que está faltando na seguinte procedure SQL?

update tablename \_\_\_\_ fieldname='value' where id=909;

# Pergunta LPI

---



Qual a palavra que está faltando na seguinte procedure SQL?

update tablename \_\_\_\_ fieldname='value' where id=909;

**Resposta: SET**

# Pergunta LPI



Qual das seguintes palavras é usada para restringir os registros que são retornados de uma consulta SELECT com base em critérios passados para os valores nos registros?

A. WHERE

B. IF

C. FROM

D. LIMIT



# Pergunta LPI

---



Qual das seguintes palavras é usada para restringir os registros que são retornados de uma consulta SELECT com base em critérios passados para os valores nos registros?

A. WHERE

B. IF

C. FROM

D. LIMIT

**Resposta: A**



[www.4LINUX.com.br](http://www.4LINUX.com.br)