UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS E PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

3 – Shell Script em Linux

Sistemas Operativos e Redes| Eng.º Milton Aguiar

Cofinanciado por:

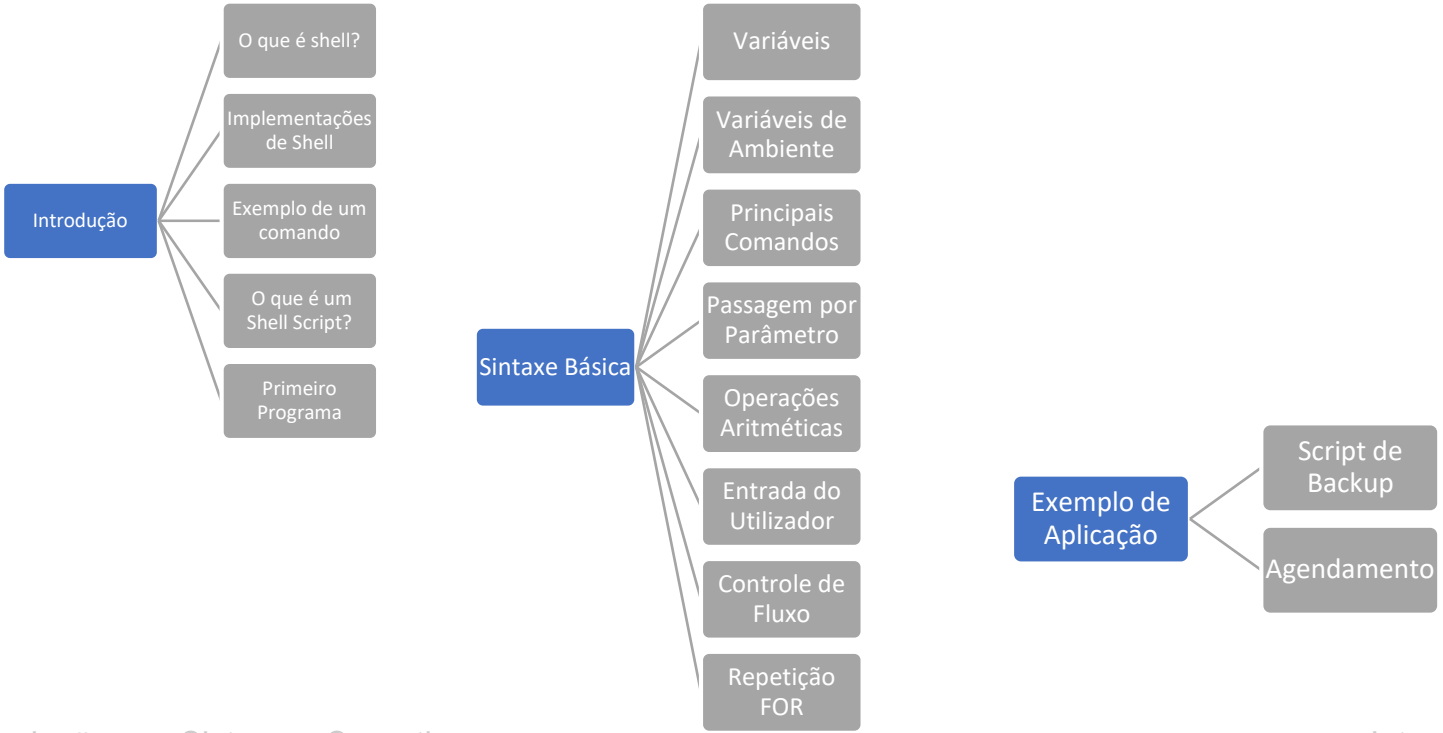
MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

Objetivos


Introdução
O que é shell?
Implementações de Shell
Exemplo de um comando
O que é um Shell Script?
Primeiro Programa
Sintaxe Básica
Variáveis
Variáveis de Ambiente
Principais Comandos
Passagem por Parâmetro
Operações Aritméticas
Entrada do Utilizador
Controle de Fluxo
Repetição FOR
Exemplo de Aplicação
Script de Backup
Agendamento

Introdução

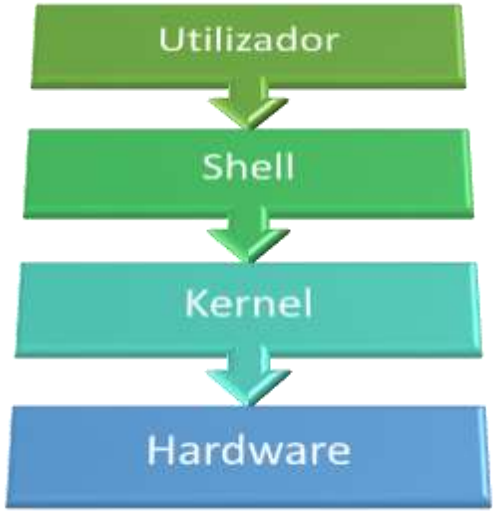
O que é Shell?

Introdução

.O que é Shell?

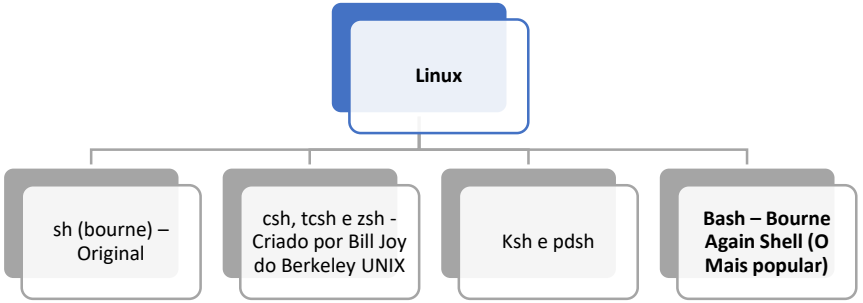
O Shell é um programa que atua na interface entre o utilizador e o kernel do sistema operativo.

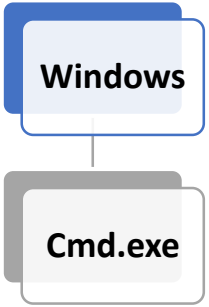
O kernel é quem acessa os equipamentos (hardware) da máquina, como disco rígido, placa de vídeo e modem.



Introdução

.Implementações de Shell

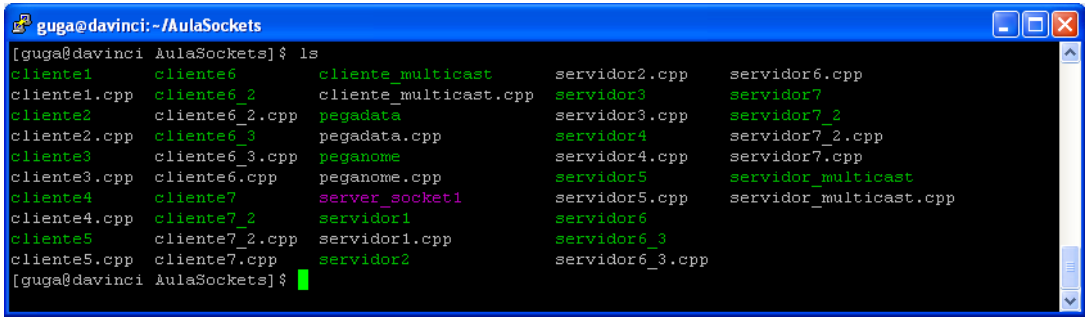




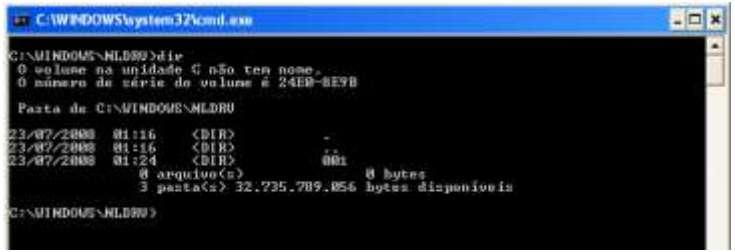
Introdução

.Exemplo de Comando

Listagem de diretório – Comando ls (Linux)



Listagem de diretório – Comando dir (Windows)



Introdução

O que é Shell Script?

Introdução

.Shell Script

Shell script é uma linguagem de programação interpretada usada em vários sistemas operativos.

De outra maneira, é uma seqüência de comandos armazenados em um ficheiro.

Ficheiro que pode ser executado.

Introdução

.Exemplos de Aplicações

Backups Automáticos

Compilar uma série de ficheiros

Criar utilizadores do sistema

Primeiro Programa!

Primeiro Programa

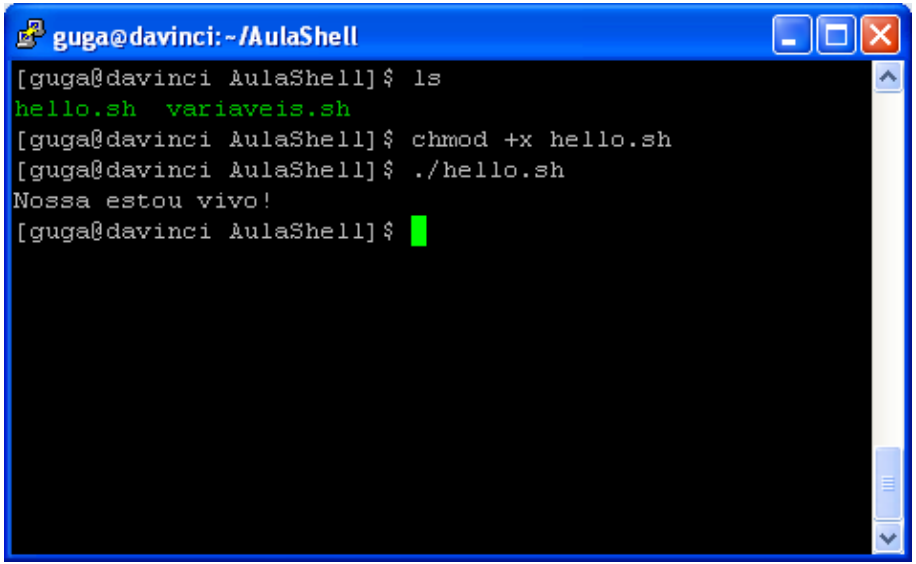
Basta criar um arquivo texto com os comandos!

A primeira linha deve ser **#!/bin/bash**

Torne-o executável com o comando **chmod**

Pronto!

```
1  #!/bin/bash
2  echo 'Nossa! Estou vivo!'
3
```

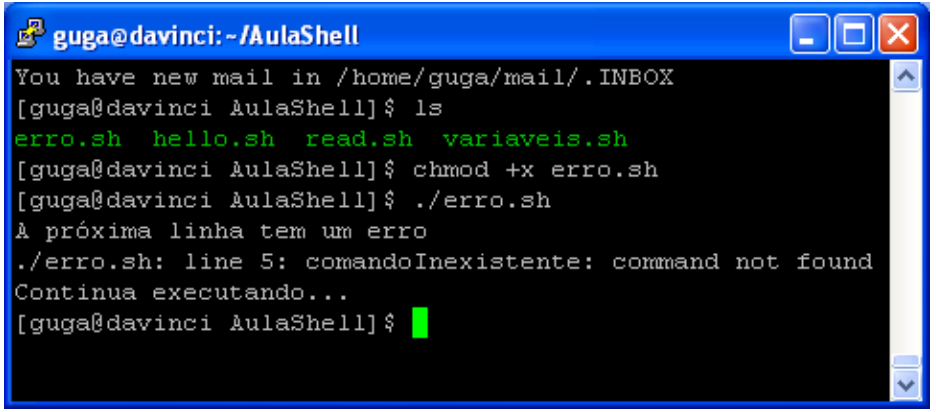


Primeiro Programa

Se um erro ocorrer, o script segue a execução dos comandos seguintes!

Comentários no código são iniciados pelo caracter #

```
1  #!/bin/bash
2
3  echo "A próxima linha tem um erro"
4  #Esse comando não existe!
5  comandoInexistente "erro"
6  echo "Continua executando..."
7
```



Sintaxe Básica Shell

Sintaxe Básica Shell

Sintaxe Básica Shell

.Variáveis

Não existe a obrigatoriedade de se declarar uma variável

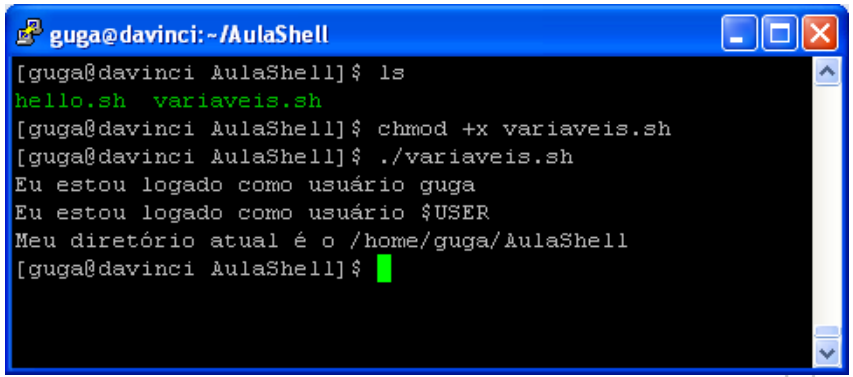
Não é preciso definir o tipo da variável

Valor pode ser uma frase, números, e até outras variáveis e comandos

Ao referenciar uma variável deve-se colocar $ antes do seu nome identificador

```
1  #!/bin/bash
2  variavel="Eu estou logado como usuário $USER"
3  echo $variavel
4  variavel='Eu estou logado como usuário $USER'
5  echo $variavel
6  variavel="Meu diretório atual é o `pwd`"
7  echo $variavel
8  #Não podem haver espaços ao redor do igual "="
9
```

Áspas duplas -> variável interpretada
Áspas simples -> valor literal
Acento grave -> interpreta comando



Sintaxe Básica Shell

.Variáveis de Ambiente

Quando o script inicia algumas variáveis de ambiente são inicializadas

Para distinguir das variáveis criadas pelo utilizador, as variáveis de ambiente são representadas com letras maiúsculas

Para ter uma lista completa das variáveis de ambiente basta digitar o comando **env**

Variável	Descrição
$HOME	O diretório HOME do utiliador corrente.
$PATH	Lista de diretórios separados por ponto e vírgula (;) onde serão procurados os comandos.
$USER	O utilizador.
$PWD	O diretório corrente.

Sintaxe Básica Shell

.Principais Comandos

Comando	Descrição
ls	Lista arquivos e diretórios **ls –a #Arquivos ocultos** **ls –l #Mais informações**
rm	Remove arquivos ou diretórios **rm -f leiame.txt** **rm -rf pasta**
mkdir	Cria um diretório **mkdir diretorio**
cp	Copia arquivos **cp manual.txt /home/manual**
mv	Move e/ou renomeia arquivos. **mv manual.txt ../** **mv manual.txt manual2.txt**
cat	Mostra o conteúdo do arquivo **cat manual.txt**
grep	Faz buscas em arquivos procurando linhas que atendas a expressão regular passada por parâmetro **grep apple fruitlist.txt** **ls | grep aula**

Mais comandos: http://www.guiaubuntupt.org/wiki/index.php?title=Comandos_basicos|

Sintaxe Básica Shell

.Passagem por Parâmetro

É possível passar parâmetros para um shell script via linha de comando

```
1  #!/bin/bash
2  echo "Número de parâmetros passados: $#"
3  echo "Nome do Script Shell: $0"
4  echo "Primeiro Parâmetro: $1"
5  echo "Segundo Parâmetro: $2"
6
```

```
guga@davinci:~/AulaShell
[guga@davinci AulaShell]$ chmod +x parametros.sh
[guga@davinci AulaShell]$ ./parametros.sh 1234 "segundo p
arâmero"
Número de parâmetros passados: 2
Nome do Script Shell: ./parametros.sh
Primeiro Parâmetro: 1234
Segundo Parâmetro: segundo parâmero
[guga@davinci AulaShell]$
```

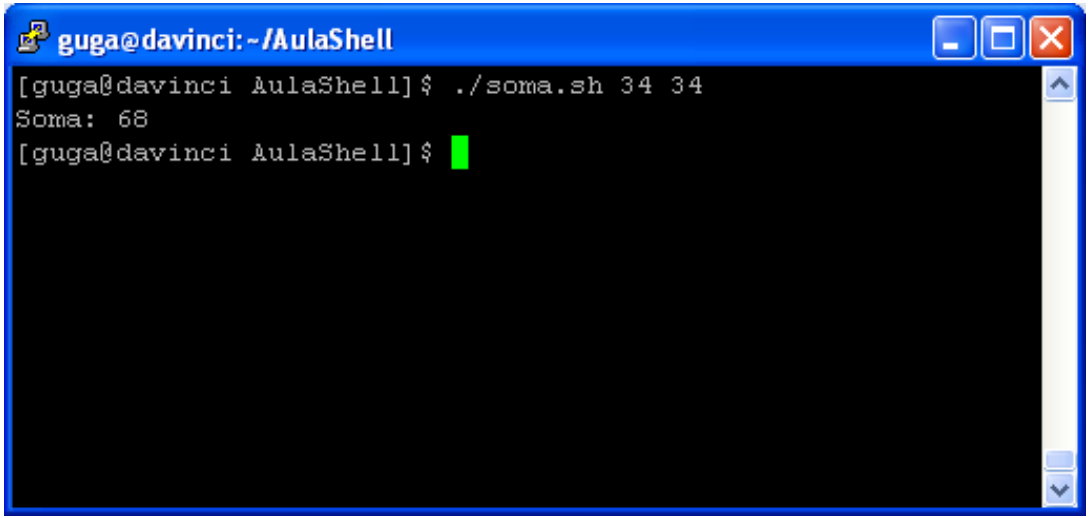
Comando	Descrição
$#	Número de Parâmetros passados
$1, $2, ...	Os parâmetros passados para o script
$0	O nome do script shell

Sintaxe Básica Shell

.Operações Aritméticas

- Formato de uma expressão aritmética: $((expressão))
- Variáveis não precisam ser precedidas de $
- Variáveis não definidas são inicializadas automaticamente com zero
- Aritmética é somente de inteiros

```
1  #!/bin/bash
2  echo "Soma: $(($1+$2))"
3
```



Sintaxe Básica Shell

.Entrada do Utilizador

Pode ser pedida a entrada de um valor para o utilizador através do comando **read**

O comando **read** bloqueia a execução do script enquando espera a entrada do valor pelo utilizador

Quando o utilizador clica <enter> a interpretação do script continua

```
1  #!/bin/bash
2
3  echo "Entre com o valor para a variável:"
4  read variavel
5  echo "O valor digitado foi: $variavel"
6
```

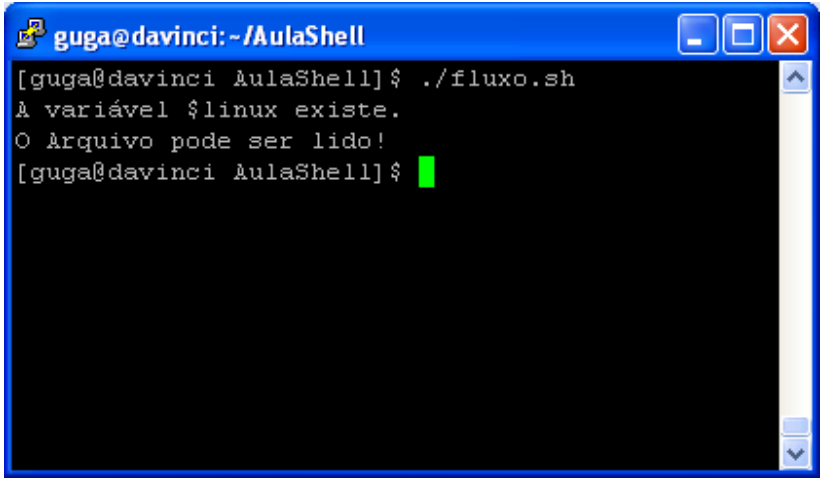
```
guga@davinci:~/AulaShell
[guga@davinci AulaShell]$ chmod +x read.sh
[guga@davinci AulaShell]$ ./read.sh
Entre com o valor para a variável:
123
O valor digitado foi: 123
[guga@davinci AulaShell]$
```

Sintaxe Básica Shell

.Controle de Fluxo (IF)

Controle de fluxo são comandos que alteram o fluxo de execução do programa de acordo com o teste de condições

```
1  #!/bin/bash
2  if [ -e $linux ]
3  then
4      echo  'A variável $linux existe.'
5  else
6      echo 'A variável $linux não existe.'
7  fi
8
9  file=leiame.txt
10 if test -r "$file"
11 then
12     echo "O Arquivo pode ser lido!"
13 else
14     echo "O Arquivo não pode ser lido!"
15 fi
16
```



Sintaxe Básica Shell

.Operadores

Operadores de Texto

strin1=string2	Testa se as strings são iguais
Strin1!=string2	Testa se as strings são diferentes
-n string	Testa se a string é não nula
-z string	Testa se a string é nula

Operadores Aritméticos

expr1 -eq expr2	Testa se as expressões são iguais
expr1 -ne expr2	Testa se as expressões são diferentes
expr1 -gt expr2	Testa se a expr1 é maior que a expr2
expr1 -ge expr2	Testa se a expr1 é maior ou igual a expr2
expr1 -lt expr2	Testa se a expr1 é menor que a expr2
expr1 -le expr2	Testa se a expr1 é menor ou igual a expr2
!expr1	Testa se expr1 é falsa

```
1  #!/bin/bash
2  echo "Você deseja executar esta
   operação? (s/n)"
3  read resposta
4  if [ "$resposta" = "s" ] ; then
5      echo "Sim o usuário deseja!"
6  else
7      echo "Não o usuário não deseja!"
8  fi
9  num1=10
10 num2=20
11 if [ $num1 -eq $num2 ]; then
12 echo "O número $num1 é igual que $num2"
13 elif [ $num1 -ge $num2 ] ; then
14     echo "O número $num1 é maior que
   $num2"
15 else
16     echo "O número $num1 é menor que
   $num2"
17 fi
18
```

Sintaxe Básica Shell

.Operadores

Operadores de Ficheiros (arquivos)

-d arq	Testa se arq é um diretório
-e arq	Testa se o arquivo existe
-f arq	Testa se arq é um arquivo regular
-r arq	Testa se a o arquivo pode ser lido
-u arq	Testa se o arquivo tem tamanho diferente de zero
-w arq	Testa se o arquivo pode ser escrito
-x arq	Teste se o arquivo pode ser executado

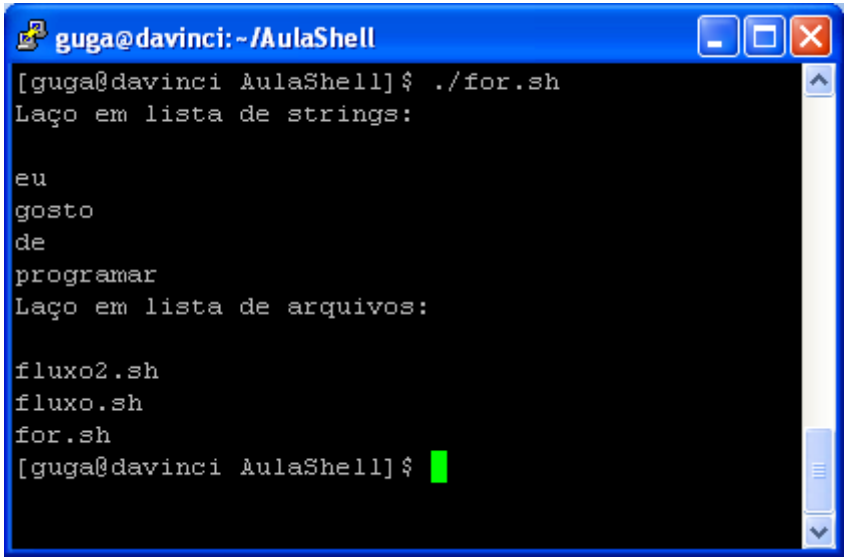
Sintaxe Básica Shell

.Repetição FOR

Em Shell Script o *for* realiza um loop em uma determinada lista de valores.

A lista pode ser um conjunto de strings, arquivos, etc..

```
1   #!/bin/bash
2
3   echo "Laço em lista de strings:"
4   echo ""
5   for  var in eu gosto de programar
6   do
7     echo $var
8   done
9
10  echo "Laço em lista de arquivos:"
11  echo ""
12  for file in $(ls f*.sh)
13  do
14      echo $file
15  done
16
```



Sintaxe Básica Shell

Exemplo de Aplicação

Exemplo de Aplicação

.Backup Automático

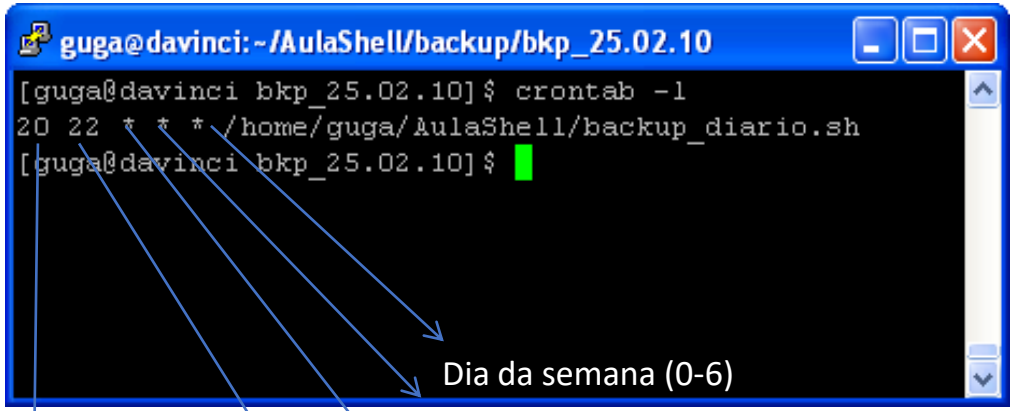
```
#!/bin/bash

dir_orig="/home/guga/AulaShell/"
dir_dest="/home/guga/AulaShell/backup"

data_dia=`date +%d`
data_mes=`date +%m`
data_ano=`date +%y`

data="$data_dia.$data_mes.$data_ano"

if [ -e $dir_orig ]; then
    echo "Diretório Origem já existente!"
else
    echo "Diretório não existe"
    exit 1
fi

dir_final="$dir_dest/bkp_$data/"

if [ -e $dir_final ]; then
    echo "Diretório Destino já existente!"
else
    echo "Diretório não existe. Será criado!"
    mkdir -p $dir_final
fi

for file in $(ls *.sh)
do
    cp $file $dir_final
done

# mensagem de resultado
echo "Seu backup foi realizado com sucesso."
echo "Diretório: ${dir_orig}"
echo "Destino: ${dir_final}";
exit 0
```

Exemplo Aplicação

.Agendando o Backup

Para agendar o backup deveremos usar o seguinte comando:

crontab -e



Campo	Função
Minuto	0-59
Hora	0-23
Dia do Mês	1-31
Mês	1-12
Dia da Semana	0-6 (O "0" é Domingo, "1" segunda, etc...)

Na internet

Comandos Linux

- **http://www.guiaubuntupt.org/wiki/index.php?title=Comandos_basicos|**

- **Moodle, Diapositivos das aulas e Internet**

Introdução aos Sistemas Operativos