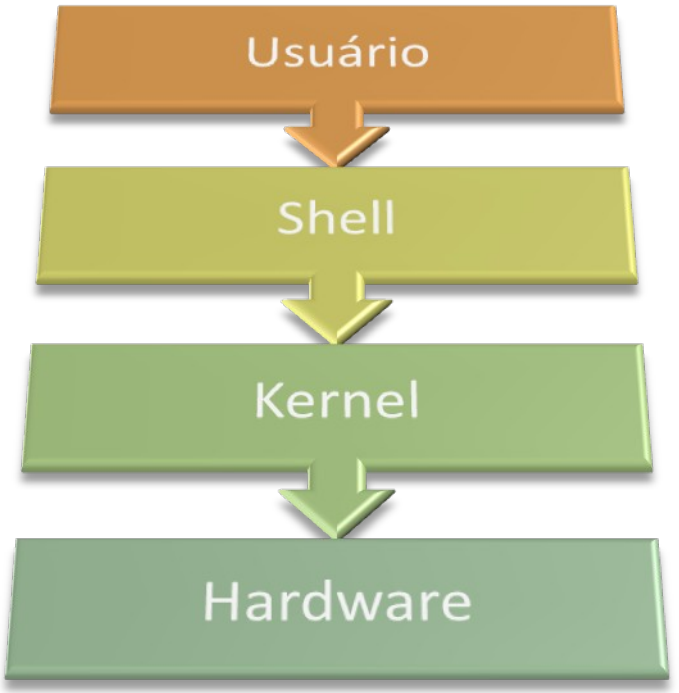
Introdução a shell scripts

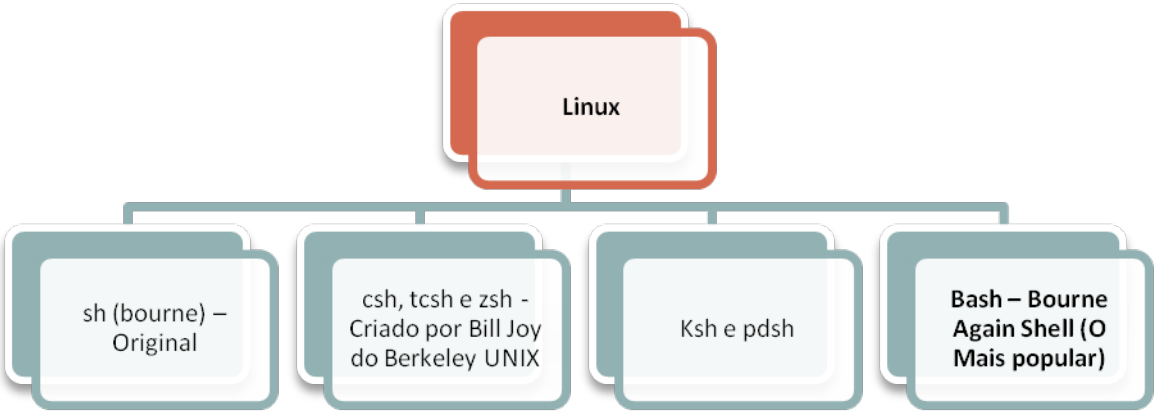
O que é?

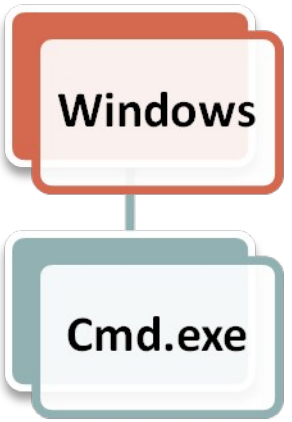
O Shell é um programa que atua na interface entre o usuário e o kernel do sistema operacional.

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

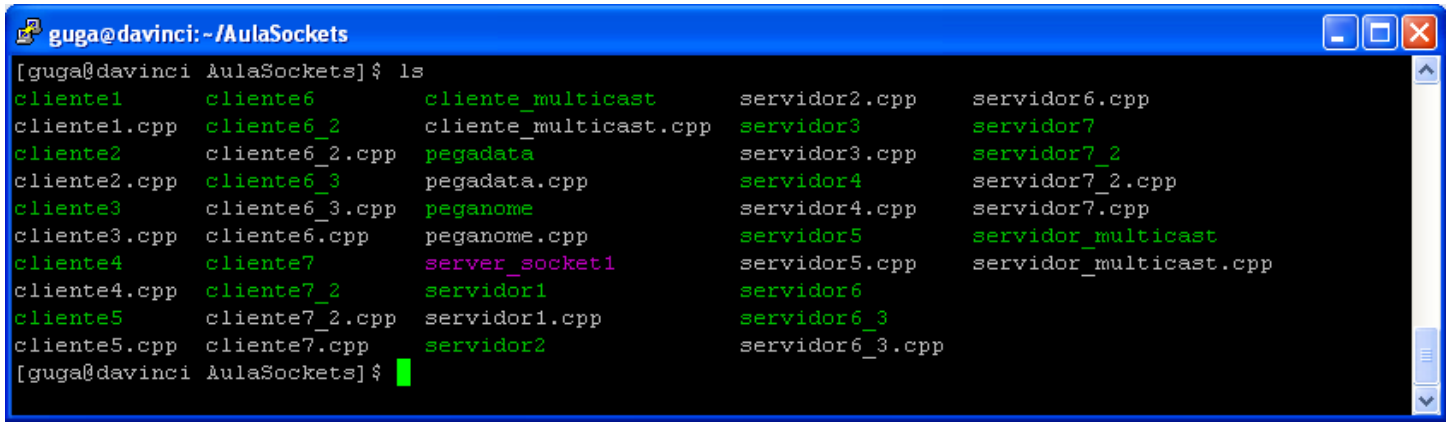


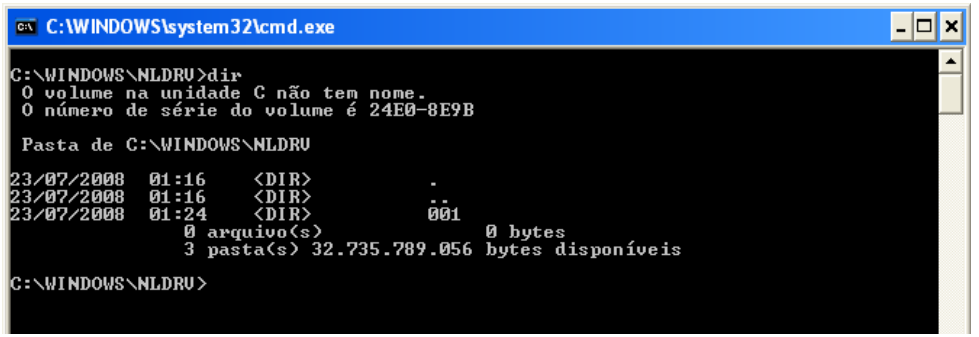
O que é


Linux
sh (bourne) – Original
csh, tcsh e zsh - Criado por Bill Joy do Berkeley UNIX
Ksh e pdsh
Bash – Bourne Again Shell (O Mais popular)


Windows
Cmd.exe

Windows e Linux





Shell script

- Um script é um arquivo que guarda vários comandos e pode ser executado sempre que preciso. Os comandos de um script são exatamente os mesmos que se digita no prompt, é tudo shell.

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 arquivo.

Arquivo pode ser executado.

Quando utilizar???

Backups Automáticos

Compilar uma série de arquivos

Criar usuários do sistema

Conceitos Iniciais

O Shell Script deverá ser utilizado sempre que for necessário realizar:

. um procedimento complexo usando muitas linhas de comando;

. um procedimento do qual todos os usuários poderão beneficiar-se;

. um comando simples usado inúmeras vezes;

. uma tarefa numa data planejada;

. integrar informações de vários sistemas existentes;

Conceitos Iniciais

- Por exemplo, se de tempos em tempos você quer saber informações do sistema como horário, ocupação do disco e os usuários que estão logados, é preciso digitar três comandos:

```
[root@localhost root]# date
[root@localhost root]# df
[root@localhost root]# w
```

Conceitos Iniciais

- É melhor fazer um script chamado "sistema" e colocar estes comandos nele. O conteúdo do arquivo "sistema" seria o seguinte:

```
#!/bin/bash
date
df
w
```

- E para chamar este script, basta agora executar apenas um comando:

  [root@localhost root]# sistema

Passos para criar um shell script

- 1. Escolher um nome para o script
- 2. Escolher o diretório onde colocar o script
  - echo $PATH
- Criar o arquivo e colocar nele os comandos
- 4. Colocar a chamada do shell na primeira linha
  - #!/bin/bash
- Tornar o script um arquivo executável
  - chmod +x sistema

Primeiro exemplo

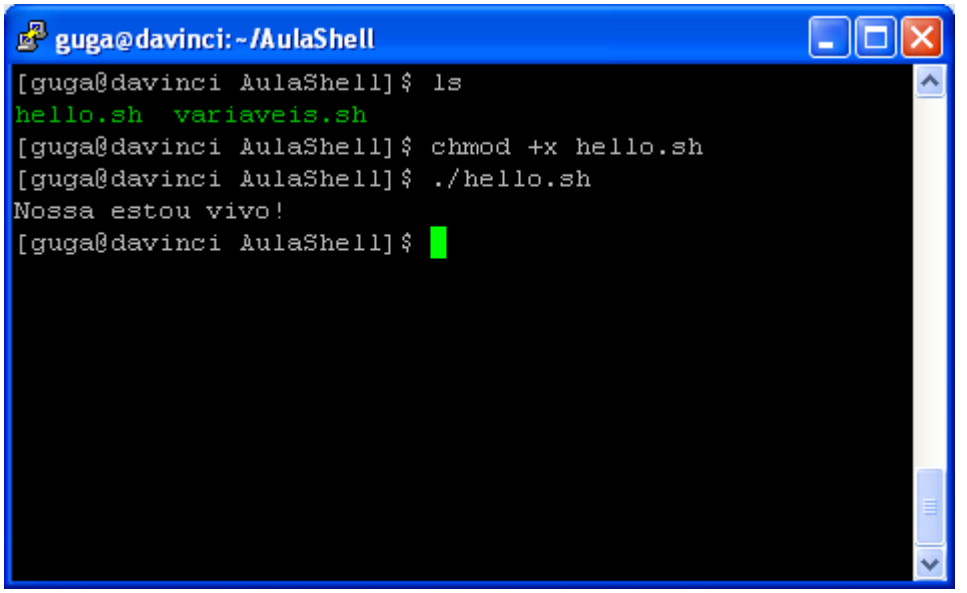
Basta criar um arquivo texto com os comandos!

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

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

Pronto!

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



Exemplo

- Crie o programa no vi com o nome info.sh, dê permissão de execução com o chmod +x e depois execute o programa com ./info.sh

```
#!/bin/bash
date
df
w
```

Melhorar a saída na tela

- Executar os três comandos seguidos resulta em um bolo de texto na tela, misturando as informações e dificultando o entendimento. É preciso trabalhar um pouco a saída do script, tornando-a mais legível.
- O comando "echo" serve para mostrar mensagens na tela.

Melhorar a saída na tela

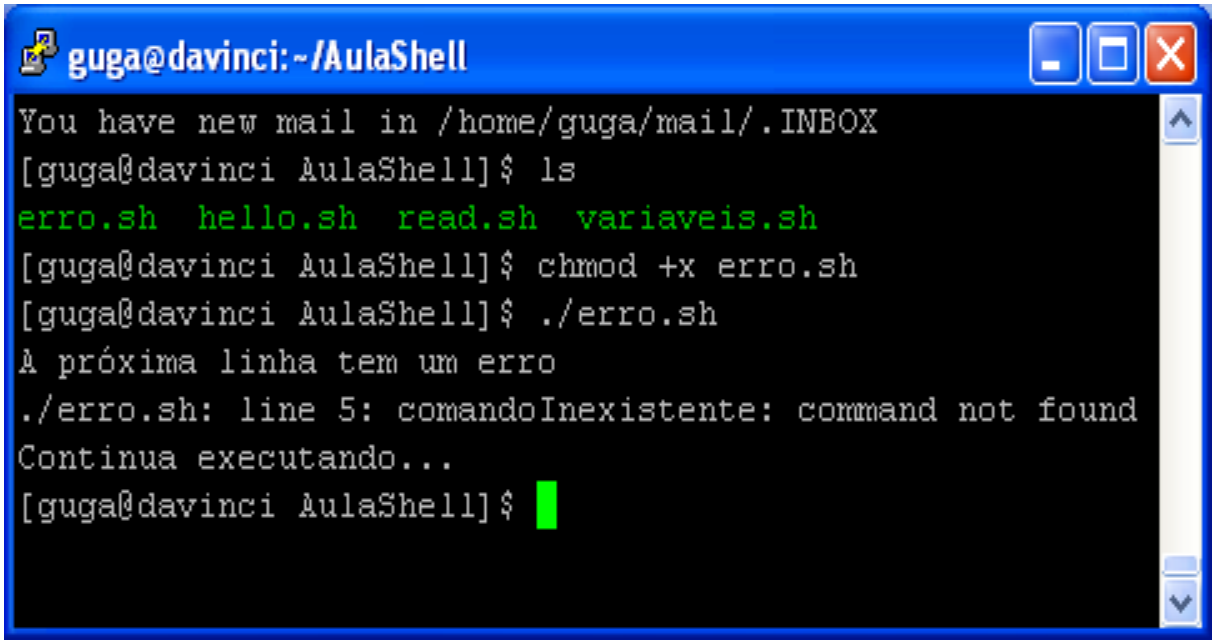
```
#!/bin/bash
echo "Data e Horário:"
date
echo
echo "Uso do disco:"
df
echo
echo "Usuários conectados:"
w
```

Erros

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

```
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
```

Comentários no código são iniciados pelo caracter #



Interagir com o usuário

```
#!/bin/bash
echo "Vou buscar os dados do sistema. Posso continuar?
[sn] "
read RESPOSTA
test "$RESPOSTA" = "n" && exit
echo "Data e Horário:"
date
echo
echo "Uso do disco:"
df
echo
echo "Usuários conectados:"
w
```

SINTAXE BÁSICA SHELL

```
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
```

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

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

Variaveis

Quando o script inicia algumas variáveis de ambiente são inicializadas

Para distinguir das variáveis criadas pelo usuário, as variáveis de ambiente são representadas com letras maúsculas

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

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

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**

Variáveis

- As variáveis são a base de qualquer script. É dentro delas que os dados obtidos durante a execução do script serão armazenados. Para definir uma variável, basta usar o sinal de igual "=" e para ver seu valor, usa-se o "echo":

```
prompt$ VARIAVEL="um dois tres"
prompt$ echo $VARIAVEL
um dois tres
prompt$ echo $VARIAVEL $VARIAVEL
um dois tres um dois tres
```

Não podem haver espaços ao redor do igual "="

Variáveis

- Ainda é possível armazenar a saída de um comando dentro de uma variável. Ao invés de aspas, o comando deve ser colocado entre "$(...)"

```
prompt$ HOJE=$(date)
prompt$ echo "Hoje é: $HOJE"
Hoje é: Sáb Abr 24 18:40:00 BRT 2004
prompt$ unset HOJE
prompt$ echo $HOJE
prompt$
```

Lembrando!!!

- Em shell é possível combinar comandos, aplicando-os em sequência, para formar um comando completo. Usando o pipe "|" é possível canalizar a saída de um comando diretamente para a entrada de outro, fazendo uma cadeia de comandos.
- Exemplo:

```
prompt$ cat /etc/passwd | grep root | cut -c1-10
```

Dica de operadores

- O conteúdo da variável é acessado colocando-se um cifrão "$" na frente
- O comando test é útil para fazer vários tipos de verificações em textos e arquivos
- O operador lógico "&&", só executa o segundo comando caso o primeiro tenha sido OK.

- O operador inverso é o "||"

testa-arquivos

Execute e interprete o seguinte script:

```
#!/bin/bash
echo -n "Digite o arquivo: "
read ARQUIVO
test -d "$ARQUIVO" && echo "$ARQUIVO é um diretório"
test -f "$ARQUIVO" && echo "$ARQUIVO é um arquivo"
test -f "$ARQUIVO" -o -d "$ARQUIVO" || echo "O arquivo '$ARQUIVO' não foi encontrado"
echo
```