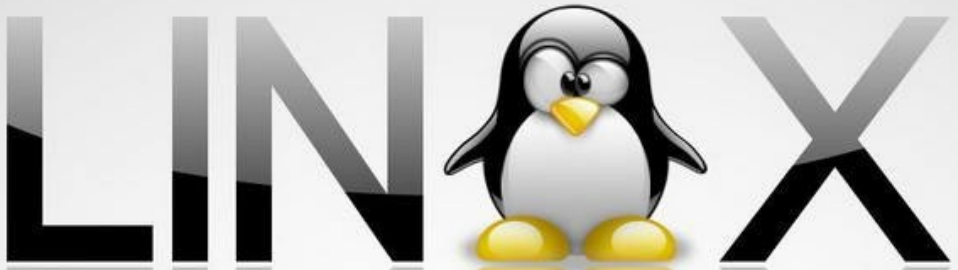
Shell script



Prof. *Lara P. Z. B. Oberderfer*

O Primeiro shell script

1. Escolha o nome para o script: **dataatual**.
2. Escolha o diretório onde ficará o arquivo: **home**
3. Crie o arquivo e insira os comandos nele: **vi data**
4. Colocar a chamada do shell na 1ª linha:

```
#!/bin/bash

date
df
w
```

5. Torne o arquivo executável: **chmod +x dataatual**

Problemas na execução

- "***Comando não encontrado***"
  O shell não encontrou o seu script.
- "***Permissão Negada***"
  O shell encontrou seu script, mas ele não é executável.
- "***Erro de Sintaxe***"
  O shell encontrou, executou seu script, porém ele contém erros.

Variáveis

- Variáveis são a base de qualquer script ou linguagem de programação.
- É nelas que os dados obtidos durante a execução do script serão armazenados.
- Para definir uma variável, basta usar o sinal de igual “**=**” e para ver o seu valor usa-se o “**echo**”
  prompt $ nome=”lara”
  prompt $ echo $nome
  lara

⚠ 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 “$(...)”, veja:

  **prompt$** HOJE=$(date)
  prompt$ echo “Hoje é: $HOJE”
  Hoje é: Ter Fev 26 15:45:00 BRT 2013
  prompt$ unset HOJE
  prompt$ echo $HOJE

Variáveis

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

  **prompt$** HOJE=$(date)

  prompt$ echo "Hoje é: $HOJE"

  Hoje é: Ter Fev 26 15:45:00 BRT 2013

  prompt$ unset HOJE

  prompt$ echo $HOJE

  apaga uma variável

Variáveis padrão

- Para ver quais variáveis que o shell já define por padrão, use o comando: “env”.

Comandos

- Sintaxe: **COMANDO OPÇÕES PARÂMETROS**
- O shell usa o *espaço em branco* para separar o comando de seus argumentos.

Comandos

Comando	*Função*	*Opções úteis*
cat	mostra arquivo	-n, -s
cut	extrai campo	-d, -f, -c
date	mostra data	-d, +"..."
find	encontra arquivos	-name, -iname, -type f, - exec
grep	encontra texto	-i, -v, -r, -qs, -w, -x
head	mostra início	-n, -c
printf	mostra texto	nenhuma
rev	inverte texto	nenhuma
sed	edita texto	-n, s/isso/aquilo/, d
seq	conta números	-s, -f
sort	ordena texto	-n, -f, -r, -k, -t, -o
tail	mostra final	-n, -c, -f
tr	transforma texto	-d, -s, A-Z a-z
uniq	remove duplicatas	-i, -d, -u
wc	conta letras	-c -w, -l, -L

Comandos: cat

- Mostra o conteúdo do arquivo.

- Mostra o script com as linhas numeradas.
- O “-n” é o número de linhas, e dataatual é o nome do arquivo

Comandos: echo

- O comando echo serve para mostrar mensages na tela.
- Para usar o echo, basta colocar o texto entre “aspas”.
- Se nenhum texto for colocado, uma linha em branco é mostrada.

Comando: read

- O comando read lê o que o usuário digita e guarda em uma variável.

Comentários

- Para inserir comentários basta iniciar a linha com # e escrever o texto em seguida. Estas linhas são ignoradas pelo shell.

Comando: test

- O comando test consegue fazer vários testes em números, textos e arquivos. Ele possui várias opções para indicar que tipo de teste será feito.

	Testes em variáveis
-lt	num é menor que (LessThen)
-gt	num é maior que (MoreThen)
-lt	num é menor igual (LessEqual)
-ge	num é maior igual (MoreEqual)
-eq	num é igual (Equal)
-ne	num é diferente (NotEqual)
=	string é igual
!=	string é diferente
-n	string não é nula
-z	string é nula

	Testes em arquivos
-d	é um diretório
-f	é um arquivo normal
-r	o arq tem permissão de leitura
-s	o tamanho do arq > 0
-w	o arq tem permissão de escrita
-nt	o arq é mais recente (NewerThan)
-ot	o arq é mais antigo (OlderThan)
-ef	o arq é o mesmo (EqualFile)
-a	E lógico (AND)
-o	OU lógico (OR)

Script que testa arquivos

prompt$ testa-arquivos

Digite o aquivo: /naoexiste

O arquivo '/naoexiste' não foi encontrado

---

prompt$ testa-arquivos

Digite o arquivo: /tmp

/tmp é um diretório

---

prompt$ testa-arquivos

Digite o arquivo: /tmp/passwd

/tmp/passwd é um arquivo

Exemplos

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

Exemplo com Interação c/ usr

```
#!/bin/bash

echo “Vou buscar dados de seu sistema. Posso continuar? [s/n]

read resposta

test “$resposta” = “n” && exit

echo “Data e Horário:”

date

echo
echo “Uso do disco:”

df

echo

echo “Usuários conectados:”

w
```

Comando: test

test “$resposta” = “n” && exit

- O comando exit foi chamado e o script foi finalizado.
- O conteúdo da variável é acessado colocando-se $ 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 “II”.

Melhorando o código do Script

- Com o tempo o script vai crescer, mais comando vão sendo adicionados, e quanto maior, mais difícil de encontrar o ponto certo onde fazer a alteração ou corrigir um erro.
- Para poupar horas de estresse, e facilitar as manutenção futuras, é preciso deixar o código visualmente mais agradável e espaçado, e colocar comentários esclarecedores.

```
#!/bin/bash
# sistema - script que mostra informações sobre o sistema
# Autor: Fulano da Silva

# Pede uma confirmação do usuário antes de executar
echo "Vou buscar os dados do sistema. Posso continuar? [sn] "
read RESPOSTA

# Se ele digitou 'n', vamos interromper o script
test "$RESPOSTA" = "n" && exit
```

```
# O date mostra a data e a hora correntes
echo "Data e Horário:"
date
Echo

# O df mostra as partições e quanto cada uma ocupa no disco
echo "Uso do disco:"
df
Echo
```

O w mostra os usuários que estão conectados nesta máquina

echo "Usuários conectados:"

w

Conceitos mais avançados

Recebimentos de opções e parâmetros

- Assim como os comandos do sistema que possuem e opções e parâmetros, os scripts também podem ser preparados para receber dados via linha de comando.
- Dentro do script, algumas variáveis especiais são definidas automaticamente, em especial, "**$1**" contém o primeiro argumento recebido na linha de comando, "**$2**" o segundo, e assim por diante.

Veja o script "argumentos":

```
#!/bin/sh
# argumentos - mostra o valor das variáveis especiais
echo "O nome deste script é: $0"
echo "Recebidos $# argumentos: $*"
echo "O primeiro argumento recebido foi: $1"
echo "O segundo argumento recebido foi: $2"
```

Veja o script "argumentos":

prompt$ ./argumentos um dois três

O nome deste script é: **./argumentos**

Recebidos 3 argumentos: um dois três

O primeiro argumento recebido foi: um

O segundo argumento recebido foi: dois

Expressões aritméticas

- O shell também sabe fazer contas. A construção usada para indicar uma expressão aritmética é "$((...))", com dois parênteses.

+ Adição

- Subtração

* Multiplicação

/ Divisão

prompt$ echo $((2*3)) 6

prompt$ echo $((2*3-2/2+3)) 8

prompt$ NUM=44

prompt$ echo $((NUM*2)) 88

**prompt$** NUM=$((NUM+1))

prompt$ echo $NUM 45

Referências Bibliográficas:

- JARGAS, Aurelio Marinho. Introdução ao Shell Script. Disponível em: http://aurelio.net/shell/apostila-introducao-shell.pdf. Acesso em: 20/02/2013.