Introdução à Operação de Sistemas Linux

Introdução a Shell Scripts

Professor:
Reinaldo Gomes
E-mail:
reinaldo@computacao.ufcg.edu.br

Sobre scripts...

- São arquivos simples contendo seqüências de comandos
  - Porém possuem estruturas de controle
    - Decisão
    - Repetição (Laços)
- Possui diversos usos
  - Instaladores
  - Rotinas de backup
  - Rotinas de administração de sistemas
- Forma fácil de automatizar tarefas
  - Interage com os comandos do Linux
  - Permite manipulação da saída dos comandos
- Linguagem Interpretada

Estrutura de um script

- Início
  - `#!/bin/bash`
  - Indica o shell utilizado para interpretar o script
- Meio
  - Comandos
    - **Qualquer comando Linux**
  - Estruturas de controle
    - **Decisões**
    - **Repetições**
  - Comentários
    - **Após o caracter #**
    - **Pode ser no início da linha ou após um comando**
    - **Texto após # é ignorado pelo interpretador**
- Fim
  - `exit 0`
  - Não é obrigatório

Primeiro script

- Crie um novo arquivo em um editor de texto qualquer
  - `nano script.sh`

- Digite os comandos do script

```
#!/bin/bash
# script imprime um texto na tela
echo Hello World
echo "Hello World"
```

  - Comando echo "ecoa" (imprime) na tela a frase passada como parâmetro acrescentando levando o cursor para a próxima linha

- Executando o script
  - Necessita de permissão de execução
    - **`chmod +x script.sh`**
  - Execução
    - **`./script.sh`**
    - **`bash /home/aula/scripts/script.sh`**

Variáveis (1)

- Variáveis não tem tipo definido
  - São strings
  - Se forem somente dígitos, bash permite operações inteiras
- Para atribuir um valor

```
#sem espaços antes e após o '='
x=world
a="Hello World"
```

- Para acessar o valor

```
y=$x #atribui "world" para y
y=x #atribui o caracter "x" para y
y=${x} #atribui "world" para y
y=${x}nn #atribui "worldnn" para y
y=$xnn #Erro (atribui "" a y)
```

- Imprimindo na tela

```
echo 'Hello $x' # Imprime Hello $x
echo "Hello $x" # Imprime Hello World
```

- Lendo uma variável do teclado

```
read -p "Digite o nome:" nome
echo $nome
```

Variáveis (2)

- Uma variável pode conter um comando

```
LS="ls"
LS_FLAGS="-l"
$LS $LS_FLAGS /home/aula/ #Lista arquivos de /home/aula
```

- Variáveis de ambiente
  - Variáveis pré-definidas pelo BASH
  - Exemplo: `$USER`, `$PWD`, `$HOME`
  - Para visualizar todas use `env`
- Passagem de parâmetros
  - Conteúdo de `script.sh`

```
#!/bin/bash
echo "$1  aula de hoje  $2"
```

  - Ao chamar o script passa-se os parâmetros

```
./script.sh Linux Shell
```

Estruturas de Decisão (1)

- O famigerado `if`
  - Básico

```
if COMANDO
then
   comando1
   comando2
   ...
else
   comandos
fi
```

  - Condição
    - Saída de qualquer comando
      - 0 é true
      - Diferente de 0 é false
    - Comando `test`

Aninhado

```
if COMANDO1
then
   comando1
   comando2
   ...
elif COMANDO2
then
   comando3
   comando4
   ...
elif COMANDO3
then
   comando5
   comando6
   ...
fi
```

Estruturas de Decisão (2)

- if/then/else
  - Comando `test`
    - Retorna o resultado da expressão
    - Sintaxe:
      - **`test operand1 operator operand2`**
      - **`[ operand1 operator operand2 ]`**
      - Atenção para os espaços entre os parâmetros
    - Exemplo

```
X=3
Y=4
if [ $X -lt $Y ]; then # $X é menor que (less than) $Y?
   echo "${X} é menor que ${Y}"
else
   echo "${X} não é menor que ${Y}"
Fi
```

Estruturas de Decisão (2)

Tabela 1 – Opções do *test* para arquivos

Opção	Verdadeiro se
-e arq	*arq* existe
-s arq	*arq* existe e tem tamanho maior que zero
-f arq	*arq* existe e é um arquivo regular
-d arq	*arq* existe e é um diretório
-r arq	*arq* existe e com direito de leitura
-w arq	*arq* existe e com direito de escrita
-x arq	*arq* existe e com direito de execução

Tabela 2 – Opções do *test* para cadeias de caracteres

Opção	Verdadeiro se:
-z cadeia	Tamanho de cadeia é zero
-n cadeia	Tamanho de cadeia é maior que zero
cadeia	A cadeia cadeia tem tamanho maior que zero
c1 = c2	Cadeia *c1* e *c2* são idênticas

Tabela 3 – Opções do *test* para números

Opção	Verdadeiro se	Significado
n1 -eq n2	*n1* e *n2* são iguais	equal
n1 -ne n2	*n1* e *n2* não são iguais	not equal
n1 -gt n2	*n1* é maior que *n2*	greater than
n1 -ge n2	*n1* é maior ou igual a *n2*	greater or equal
n1 -lt n2	*n1* é menor que *n2*	less than
n1 -le n2	*n1* é menor ou igual a *n2*	less or equal

Estruturas de Decisão (3)

- case
  - Permite múltiplas opções de decisão
  - Estrutura

```
case "$var" in
 "$cond1" )
 comandos...
 ;;
 "$cond2" )
 comandos...
 ;;
esac
```

  - Atenção para a variável `$var` entre aspas

Estruturas de repetição (1)

- for
  - Exemplos

```
for planeta in Mercúrio Vênus Terra Marte
do
  echo $planeta
done
```

```
for i in {1..10}
do
   echo $i
done
```

```
for ((i=1; i <= LIMIT ; i++))
do
  echo $i
done
```

Estruturas de repetição (2)

- while
  - Exemplo
    - Uso do `test` para checar o conteúdo da variável

```
while [ "$var" != "end" ]
do
   read -p "Digite um nome: (end para sair)" var
   echo "Nome = $var"
done
```