Programação Shell Script

REDES DE COMPUTADORES

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Construções da Linguagem

* If...else...fi
  * Se a condição if for verdadeira o script irá executar o bloco verdadeiro (then) senão irá executar o bloco falso (else).

Sintaxe:

if *condição*
then
condição é zero (verdadeiro - 0)
executa todos os comandos até o inicio do bloco else
else
se a condição for falsa
executa todos os comandos até o fi
fi

Exercício

* Desenvolva um Shell Script para verificar se um ***argumento é positivo ou negativo***
* Logo após, tente implementar uma função nesse script para ***verificar se o usuário realmente digitou algum argumento***.
  * Se o usuário não tiver digitado nenhum argumento, informe-o que ele deve digitar pelo menos um argumento

```
#!/bin/sh
#
# Script para verificar se um argumento eh positivo ou negativo
if [  $# -eq 0  ]
then
echo  "$0 : Vc deve fornecer um valor como argumento
   ./<script> <argumento>"
exit 1
fi

if  test  $1  -gt  0
then
echo  "$1 eh positivo"
else
echo  "$1 eh negativo"
fi
```

Entendendo o Script

1º - O primeiro if verifica ***o numero de argumentos*** ($#) passados para o script. Se ele for igual a zero quer dizer que o usuário não passou nenhum argumento

* $0 - Pega o nome do seu próprio script e mostra na mensagem
* O exit 1 - o seu programa sai e configura o status de saída para 1, ou seja, indica que o seu script não foi executado corretamente

Entendendo o Script

2º - O segundo if verifica se ***o valor digitado como argumento é maior ou menor do que zero***. Será executado apenas se o primeiro if verificar que possui um argumento

* Se verdade, argumento maior do que 0, então mostra mensagem eh positivo
* Se falso, então mostra a mensagem eh negativo

Encontrando o erro

* Esse script **não** está logicamente correto. Por que?

<<Ponto Extra>>

Entendendo o Script

* Teste o script com o valor zero (0)
  * Resp: **0 eh negativo**

Zero não faz parte dos numero negativos.

Entendendo o Script

* Como resolver este problema?

Entendendo o Script

* Para ***resolver este problema*** altere a condição do segundo if para “if test $1 **-ge** 0”

```
#!/bin/sh
#
# Script para verificar se um argumento eh positivo ou negativo
if [  $# -eq 0  ]
then
echo  "$0 : Vc deve fornecer um valor como argumento
   ./<script> <argumento>"
exit 1
fi

if  test  $1  -ge  0
then
echo  "$1 eh positivo"
else
echo  "$1 eh negativo"
fi
```

Construções da Linguagem

* If...else...fi **aninhados** (um dentro do outro)

* Podemos escrever uma estrutura:

if ... else ... fi completa,

dentro do bloco else de um if

Isso é o que chamamos de if's aninhados.

If...else...fi aninhados

* Exemplo

```
#!/bin/sh
# Nome: ifaninhados.sh
opcao=0
echo "1. Unix (Sun Os)"
echo "2. Linux (Red Hat)"
echo -n "Selecione o Sistema Operacional [1 ou 2]:  "
read opcao

if [ $opcao  -eq  1 ] ; then
    echo "Voce escolheu o Unix (Sun Os)"
else          #### if aninhado ######

     if [ $opcao  -eq  2 ] ; then
          echo "Voce escolheu o melhor - Linux (Red Hat)"
     else
          echo "Voce nao gosta de Sistemas Unix Like"
     fi
fi
```

If...else...fi aninhados - Sintaxe

```
if   condição
then
   if   condição
   then
            .....
            faça isso
   else
            ....
            faça isso
   fi
else
   .....
   faça isso
fi
```

Construções da Linguagem

* If...then...else **multi-níveis**

* Sintaxe:

```
if   condição
        then
                condição é zero (true - 0)
                executa todos os comandos até o elif
        elif   condição1
        then
                condição1 é zero (true - 0)
                executa todos os comandos até o elif
        elif   condição2
        then
                condição2 é zero (true - 0)
                executa todos os comandos até o else
        else
                #Nenhuma das condições acima
                #condição,condição1,condição2 são verdadeiras
                #(todas acima são falsas, diferentes de zero)
                #executa todos os comandos até o fi
fi
```

Exemplo

* Digite o script a seguir:

```
#
#!/bin/sh
# Script para testar if..elif...else
#
if [ $1 -gt 0 ]; then
  echo "$1 eh positivo"
elif [ $1 -lt 0 ]
then
  echo "$1 eh negativo"
elif [ $1 -eq 0 ]
then
  echo "$1 eh zero"
else
  echo "Opps! $1 nao eh um numero, forneca apenas
   numeros"
fi
```

Loops

* Loops em Shell Script

Loops

* Definição:
  * O computador pode ***repetir uma determinada instrução*** quantas vezes for necessário até que se satisfaça uma condição em particular.
  * Um ***grupo de instruções*** executado repetidamente é chamado de loop

* O bash suporta:
  * Loop **for**
  * Loop **while**

Loops

Atenção! As condições abaixo devem ser atendidas para a execução dos loops (While / For)

a) A variável utilizada na condição do loop ***deve ser inicializada*** para que a execução comece

b) Um teste (***condição***) é feito no início de cada iteração do loop

c) O corpo do loop ***termina com a alteração do valor da variável de teste*** (da condição)

Loop - for

* Loop - for

Loop - for

Sintaxe:

for { ***nome da variável*** } **in** { ***lista*** }

do

- executa uma vez para cada item que estiver na lista até ela terminar e
- repete todos os comandos entre ***do*** e ***done*** a cada iteração

done

Loop - for

Exemplo:

```
#!/bin/sh
# ----------------------------------------
#Script para testar o for
# ----------------------------------------

for   i   in   1 2 3 4 5
do
  echo   "BemVindo $i vez(es)"
done
```

Loop - for

* Exemplo 2

```
#!/bin/sh
#Script 2 para testar o for

if [ $# -eq 0 ]
then
   echo "Erro - Faltando argumento na linha de comandos"
   echo "Sintaxe: $0 numero"
   echo "Utilize para saber a tabuada de multiplicacao do numero fornecido"
exit 1
fi

n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
   echo "$n * $i = `expr $i \* $n`"
done
```

Loop - for

* Outra implementação para o laço de repetição (loop) for:

Sintaxe:

for ((***expr1***; ***expr2***; ***expr3***))

do

repete todos os comandos entre o **do** e o **done** até que a ***expr2*** seja verdadeira (TRUE)

done

Loop - for

* Entendendo como funciona este tipo de implementação do for:

```
#!/bin/bash
for ((  i = 0 ;  i <= 5;  i++  ))
do
        echo "BemVindo $i vez(es)"
done
```

Loop - for

* Vimos anteriormente que o loop if pode ser aninhado dentro de outro.

* **O loop for também pode ser aninhado?**

Loop – for aninhado

* Loop for **aninhado**

* Similarmente ao loop if, nós podemos aninhar loops for dentro de outros loops for.

  * Para entender como isso pode ser feito, veja o exemplo a seguir:

```
#!/bin/bash
### Loop for externo ###

for (( i = 1; i <= 5; i++ ))
do
        ### Loop for interno ###
        for (( j = 1 ; j <= 5; j++ ))
        do
                echo -n "$i "
        done

echo "" #### imprime uma nova linha ###

done
```

Loop – for aninhado

* Rode o script anterior e veja o resultado:

* 1 1 1 1 1
* 2 2 2 2 2
* 3 3 3 3 3
* 4 4 4 4 4
* 5 5 5 5 5

Loop – for aninhado

* Entendendo o resultado:

* Para cada valor de i do ***loop for externo***, o j do ***loop for interno*** executa 5 vezes (de 1 a 5) repetindo a mensagem com o valor de i na tela;

* O programa termina quando o valor de i atinge seu limite, 5.

Loop – for aninhado

* Vamos ver algo interessante...
* Digite o script a seguir e veja o resultado!

```
#!/bin/bash
### Loop for externo ###
for (( i = 1; i <= 9; i++ ))
do
   ### Loop for interno ###
   for (( j = 1 ; j <= 9; j++ ))
   do
        tot=`expr $i + $j`
        tmp=`expr $tot % 2`
        if [ $tmp -eq 0 ]; then
                        echo -e -n "\033[47m "
        else
             echo -e -n "\033[40m "
        fi
  done
#### configura a cor de fundo para preto
echo -e -n "\033[40m"
#### imprime uma nova linha ###
echo ""
done
```

Loop – while

* O loop *while*
* O loop while ***executa enquanto sua condição de teste for verdadeira***

Sintaxe:

```
while [ condição ]
do
    comando1
    comando2
    comando3
    ....
done
```

Loop – while

* Para entender o loop **while**, voltaremos ao script apresentado anteriormente no loop for:

```
#!/bin/sh
#Script 2 para testar o for
if [ $# -eq 0 ]
then
   echo "Faltando argumento na linha de comandos"
   echo "Sintaxe: $0 numero"
   echo "Utilize para saber a tabuada de multiplicacao
   do numero fornecido"
exit 1
fi

n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
   echo "$n * $i = `expr $i \* $n`"
done
```

Loop – while

* Esse mesmo script *pode ser reescrito* utilizando o *loop while*:

```
#!/bin/sh
#Script 2 para testar o for
if [ $# -eq 0 ]
then
   echo "Faltando argumento na linha de comandos"
   echo "Sintaxe: $0 numero"
   echo "Utilize para saber a tabuada de multiplicacao
   do numero fornecido"
exit 1
fi

n=$1
i=1    ###atencao###
while [ $i -le 10 ]
do
   echo "$n * $i = `expr $i \* $n`"
   i=`expr $i + 1`    #atencao
done
```

O comando case (caixa de opções)

* O comando case é uma boa alternativa na ***substituição*** do ***if-then-else-fi*** de vários níveis

* Ele nos possibilita ***associar diversas alternativas*** para um valor passado apenas para uma variável

* Comando Case – Sintaxe:

```
case $nome-variavel in
        padrao1)  comando
                        ...
                  comando;;
        padrao2)  comando
                        ...
                  comando;;
        padraoN)  comando
                        ...
                   comando;;
        *) comando
              ...
           comando;;
esac
```

O comando case (caixa de opções)

* Exemplo:

```
#!/bin/bash

if [ -z $1 ]  #se argumento for nulo
then
  aluguel="*** Veiculo Desconhecido ***"
elif [ -n $1 ]
then
  ### senao utilize o argumento passado ###
  aluguel=$1
fi

case $aluguel in
   "carro") echo "Para alugar o $aluguel Rs.10,00 por Km";;
   "van") echo "Para alugar o $aluguel Rs.20,00 por Km";;
   "jeep") echo "Para alugar o $aluguel Rs.30,00 por Km";;
   "bicicleta") echo "Para alugar o $aluguel Rs.5,00 por Km";;
   *) echo "Desculpe, nos nao temos $aluguel para alugar
    para voce";;
esac
```

Debug

* Você se lembra do comando sh para rodar scripts sem alterar as permissões...

* Vamos atribuir a ele outra funcionalidade
  * **Debugger**

* Para ***debugar*** o código do script anterior execute a seguinte linha

sh -x <nome> <argumento>

Obrigado.

Contato: joaopauloaramuni@gmail.com

UNIVERSIDADE FUMEC

50 Anos