Introdução à Ciência da Computação

Shell Script – parte IV

Professor Iago Augusto de Carvalho

iago.carvalho@unifal-mg.edu.br

1

Teste de condições compostas

A declaração if-then permite usar lógica booleana para combinar testes. Podemos usar dois operadores booleanos:

AND

[ condição1 ] && [ condição2 ]

OR

[ condição1 ] || [ condição2 ]

2

Testar de comparações compostas

#!/bin/bash

#Testar comparações compostas.

#Vamos verificar se o usuário logado atualmente é adriana #e se ela tem permissão de escrito no seu arquivo .bashrc

if[ $USER = adriana ] && [ -w $HOME/.bashrc ]

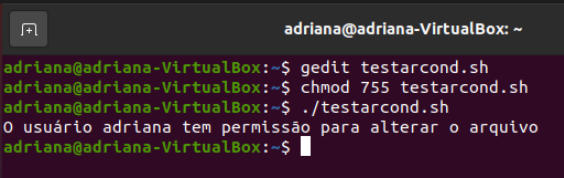
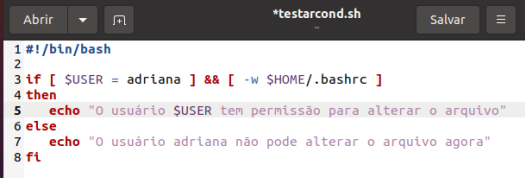
then

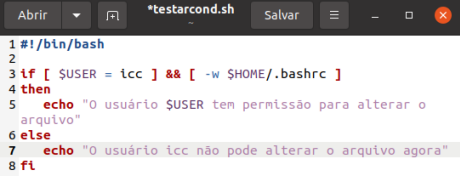
echo “O usuários $USER tem permissão para alterar o arquivo” else

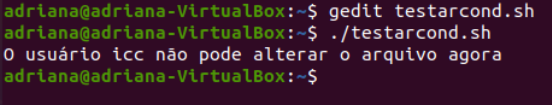
echo “O usuário adriana não pode alterar o arquivo agora” fi

3

Testar de comparações compostas

4

Testar de comparações compostas 



5

Comando case

Esse comando verifica condições múltiplas que podem ocorrer em uma variável, em um formato parecido com uma lista.

A estrutura case substitui e simplifica o uso do if-then elif com várias declarações.

O comando case compara o valor de uma variável ou expressão com os valores da lista criada.

Se o valores forem iguais, o shell executará os comandos especificados para o valor.

**Comando case – sintaxe**

case <variável> in

valor1)

comandos 1;;

valor2)

comandos 2;;

valor3 | valor4)

comandos 3 e 4;;

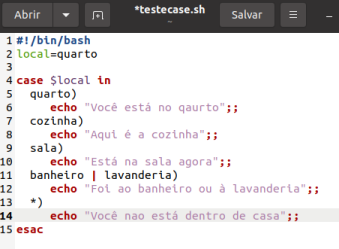
\*)

comandos-padrão;;

esac

6

Comando case

#!/bin/bash 

#Usando o comado case

local=quarto

case $local in

quarto)

echo “Você está no quarto;;

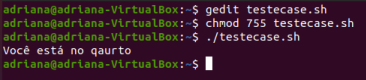
cozinha)

echo “Aqui é a cozinha;;

sala)

echo “Está na sala agora;;

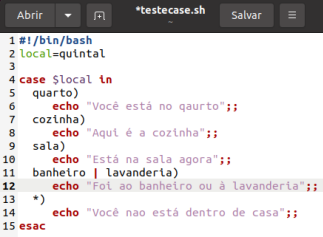
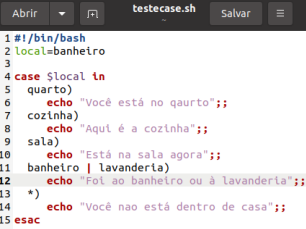
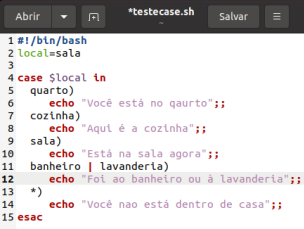
banheiro | lavanderia)

echo “Foi ao banheiro ou à lavanderia;; \*) 

echo “Você não está dentro de casa;; esac

7

Comando case



8

Estruturas de Repetição

O shell bash oferece três

estruturas de repetição para

criação de loops estruturados.

São:

for while until

9

Comando for

O comando for permite criar um loop que itera através de uma série de valores. Cada iteração executa um conjunto definido de comandos usando um dos valores da lista.

Sintaxe básica:

**for *valor* in *lista***

**do**

**comandos**

**done**

A cada iteração, a variável valor contem o valor atual da lista

10

Comando for

#!/bin/bash

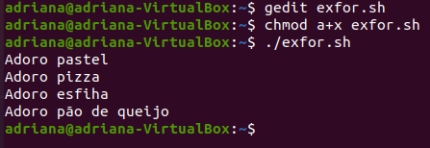
#Ler valores de uma lista, declarada no próprio comando for

for valor in pastel pizza esfiha ‘pão de queijo’

do

echo Adoro $valor

done

11

Separador de campos: IFS

Caso os itens iterados sejam compostos por palavras separadas por espaço (como Pão integral), o comando for irá considerar cada palavra como um item separado.

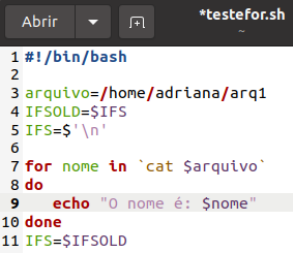
Isso é devido á variável de ambiente IFS (Internal Field Separator), a qual define uma lista de caracteres que o shell bash usa como separadores de campos.

São eles, por padrão: espaço, tabulação e newline.

É possível alterar essa lista de separadores.

12

Comando for

#!/bin/bash 

#Arquivo arq1 contém uma lista de nomes, # um por linha,

#incluindo nomes compostos

arquivo=/home/adriana/arq1

IFSOLD=$IFS

IFS=$’\n’

for nome in `cat $arquivo`

do

echo “O nome é: $nome”

done 

IFS=$IFSOLD

13

Comando for

14

Comando for

15

Comando for

#!/bin/bash

#Iterando por todos os itens de um diretório 

for item in /home/adriana/\*

do

if [ -d “$item” ]

then

echo “O item $item é um diretório” elif [ -f “$item”]

then

echo “O item $item é um arquivo” fi

done

16

Comando for



17

Comando while

O comando while permite definir um comando a testar e então iterar por um conjunto de comandos enquanto o comando definido de teste retornar status de saída zero.

Quando o comando de teste retornar status de saída diferente de zero, o while para de executar seu bloco de comandos e o loop é encerrado.

Sintaxe:

**while comando\_de\_teste**

**do**

**bloco de comandos**

**done**

O comando\_de\_teste usa o mesmo formato da estrutura if-then, e podemos usar o comando test para testar condições.

O status de saída do comando de teste deve mudar em algum momento durante as iterações, ou teremos um loop infinito.

18

Comando while

#!/bin/bash 

#Testando o comando while

var=100

while [ $var -gt 0 ]

do

echo $var

var=$[ $var -1 ]

done

19

Comando until

O comando until opera de forma oposta ao comando while.

É necessário especificar um comando de teste que retorne um status de saída diferente de zero para que o bloco de comandos listado no loop seja executado.

Quando o comando de teste retornar status de saída zero, o loop termina. Sintaxe:

**until comando\_de\_teste**

**do**

**bloco de comandos a executar**

**done**

20

Comando until 

#!/bin/bash

#Teste da estrutura de repetição until var=50

until [ $var -eq 0]

do

echo $var

var=$[ $var - 2 ]

done



21

Comando for no estilo Linguagem C

É possível usar uma estrutura de repetição for no estilo da linguagem C em um script do shell. Neste caso, teremos uma variável contadora que irá controlar o número de iterações do loop.

Sintaxe:

**for (( atrib\_variável; condição; processo\_iteração ))**

Exemplo:

**#!/bin/bash**

**#Exemplo de comando for no estilo linguagem C**

**for (( i = 1; i <= 15; i++ ))**

**do**

**echo “Número: $i”**

**done**

22

Comando for no estilo Linguagem C 23

Manipulação de entradas de usuários

Frequentemente, precisamos escrever scripts que interajam com o usuário.

Há algumas formas de se obter dados dos usuários no shell, como parâmetros de linhas de comando, opções e leitura de dados diretamente do teclado.

**Parâmetros de linha de comando**

É o método para passar dados ao script do shell. Os parâmetros de linha de comando permitem adicionar valores de dados à linha de comandos ao executar o script.

Para passar parâmetros a um script, digite-os após o nome do script, ao executá-lo: **./scriptnome param1 param2 ... paramN**

24

Leitura dos parâmetros

O shell bash atribui variáveis especiais, denominadas parâmetros posicionais, a todos os parâmetros digitados na linha de comandos.

Os parâmetros posicionais são números, sendo $0 o nome do programa, $1 o primeiro parâmetro, $2 o segundo, e assim por diante até o nono, que é $9.

Para adicionar mais parâmetros, englobe o número do parâmetro entre parênteses: $(10)

Exemplo:

**#!/bin/bash**

**#Teste de parâmetros na linha de comandos**

**echo “Programa que calcula o quadrado de um número”**

**quadrado=$[ $1 \* $1 ]**

**echo “O quadrado de $1 é $quadrado**”

Para execuitá-lo e calcular o quadrado de 3:

./quadrado.sh 3

25

Leitura dos parâmetros





26

Variáveis de parâmetros especiais

Há algumas variáveis especiais disponíveis no shell bash.

A variável especial **$#** contém o número de parâmetros de linhas de comando fornecidos ao rodar o script. Podemos usá-la para verificar se o usuário digitou o número de parâmetros necessários para rodar o programa corretamente.

Exemplo:

#!/bin/bash

#Verificação de número de parâmetros

if [ $# -ne 1 ]

then

echo “Digite ao menos um valor!”

else

resultado=$[ $1 \* 3 ]

echo “O triplo de $1 é $resultado”

fi

27

Variáveis de parâmetros especiais

28

Comando shift (deslocamento)

O comando shift auxilia na manipulação de parâmetros de linha de comando. O comando shift desloca os parâmetros em suas posições relativas.

Ao ser usado, o comando shift diminui cada parâmetro em uma posição, de modo que o valor da variável $3 é movido para $2, o valor de $2 é movido para $1 e o valor de $1 é descartado.

Exemplo:

#!/bin/bash

#Teste de comando shift

i=1

while [ -n “$1” ]

do

echo “O parâmetro $i tem o valor: $i”

i=$[ $i + 1 ]

shift

done

29

Comando shift (deslocamento)

30

Trabalhos com opções

Opções são letras precedidas por um hífen que alteram o comportamento de um comando. Para processar opções passadas ao script, vamos usar uma combinação dos comandos case e shift.

Exemplo: 

#!/bin/bash

#Usando opções em um script

while [ -n “$1” ]

do

case “$1” in

-a) echo “Opção selecionada: A”;;

-b) echo “Opção selecionada: B”;;

-c) echo “Opção selecionada: C”;;

-\*) echo “Opção incorreta”;;

esac

shift

done

31

Trabalhos com opções

32

Referências

PRITCHARD, S.; PESSANHA, B. G.; LANGFELDT, N.; STANGER, J.; DEAN, J. 2007. **Certificação Linux LPI Rápido e Prático. Guia de Referência nível 1: Exames 101 e 102**. 2ª Ed. Rio de Janeiro: Editora Alta Books.

**Curso de Shell Scripting – Bóson Treinamentos**

http://www.bosontreinamentos.com.br/curso-de-shell-scripting/

33