**PROGRAMANDO EM SHELL SCRIPT**

Exemplos de código:

- Criar e conceder permissões ao script

*touch teste.sh*

*chmod 755 teste.sh*

*vi teste.sh*

O trecho cria um arquivo teste.sh, em seguida concede a permissão e após, abre com o vim. (o *vi teste.sh* também pode ser usado para criar e abrir o arquivo.

- Editando o script

***i para entrar no modo edição do vim***

*#!/bin/bash*

*echo Hello World*

***esc para sair da edição***

***:wq para salvar e sair do vi***

Aqui, já dentro do vim, entramos no modo edição e a primeira linha do programa deve indicar o interpretador de comandos a ser utilizado, neste caso o **bash**

- Executando o script

*./teste.sh*

Simples assim! Observar a indicação do caminho do arquivo *./*

*­*- Condicional ***test***

*#Linha de comentário*

*test 1=1; echo Primeira forma de uso*

*[ 1 = 1 ]; echo Segunda forma de uso*

E então, # denota um comentário de linha e a condicional *test* pode ser usada destas duas formas mostradas. Ele irá usar os operadores de comparação.

- Condicional ***if then elif then else fi***

*echo “Digite um numero: ”*

*read numero*

*if [ $numero –gt 10 ]*

*then*

*echo “O numero é maior que 10”*

*elif [ $numero –lt 5 ]*

*then*

*echo “O numero é menor que 5”*

*else*

*echo “O numero está entre 5 e 10”*

*fi*

O *if* é sempre seguido de um *then*. E dentro dos [ ] o espaço é importante. Na verdade o *if*  testa uma saída lógica. O *elif* é um else if e também é seguido de um *then.* Já o else não precisa. E a estrutura termina com *fi*, que é o if invertido, assim como outras ocorrências dessa peculiaridade.

*echo* é um comando de saída de dados na tela e *read* de entrada de dados pelo teclado (*read –t4* é uma flag de timeout do read *-n1* le um caractere sem enterm *-s* para senhas)

- Condicional ***case***

*case $numero in  
 1)  
 echo “numero 1”;;*

*2)  
 echo “numero 2”;;  
 \*)  
 echo “Default”;;  
 esac*

OBS: o final de cada condição precisa de ;; A cada condição tem um ) . O Default é \*) . Fecha com *esac*

- Laço de repetição ***for***

*for variável in 1 2 3 4 5*

*do*

*echo Variavel:$variavel*

*done*

Essa é a primeira forma básica de se implementar um laço for, funciona assim: Os comandos do bloco serão executados conforme a quantidade de itens do array após o *in*, a variável vai ser escrita com cada índice do array. Cada índice é separado por um espaço em vazio.

*for (( i = 1 ; i <= 5 ; i++ ))*

*do*

*echo Variavel$i*

*done*

Essa implementação do for é semelhante ao que se usa na linguagem C.

- Laço de repetição ***while*** *variavel=0*

*while [ $variavel -lt 5 ]  
 do*

*echo $variavel  
 let variavel=$variavel+1*

*done*

O ***while*** testa a condição enquanto o status de saída retorna 0 (sucesso). O ***until*** funciona com a lógica inversa, até que o status de saída seja 0

- Laço de repetição ***until*** *variavel=0*

*until [ $variavel -ge 5 ]  
 do*

*echo $variavel  
 let variavel=$variavel+1*

*done*

- Funções

*minhaFuncao()*

*{  
 echo Primeira função  
 }*

*minhaFuncao*

É simples a declaração de uma função nas 4 primeiras linhas e a chamada na última linha, e ainda existe mais uma forma;

*function novaFuncao*

*{  
 echo Segunda função*

*}*

*novaFuncao*

O que muda é o uso da palavra ***function*** antes do nome e não precisa dos ***( )***

As funções devem ser declaradas antes de serem chamadas

- Cálculos matemáticos exetutados diretamente no bash são possíveis apenas para números inteiros.

*var1=5  
 var2=5  
 var3=$[$var1+$var2]  
 var4=$[4+4]*

*echo Var1 + Var2 = $var3  
 echo 4 + 4 = $var4*

*echo $var3 dividido por $var4 = $[$var3/$var4] Por que o calculo só considera numeros inteiro*

Observe que para efetuar o cálculo, deve ser entre $[ ... ] e não é dispensado o uso do $ dentro do [ ] para referênciar uma variável

Para cálculos mais precisos podemos usar a calculadora do shell ( BC )

*var1=5  
 var2=5  
 var3=$[$var1+$var2]  
 var4=$[4+4]*

*echo Var1 + Var2 = $var3  
 echo 4 + 4 = $var4*

*echo “O resultado de $var3 / $var4 é `echo “scale=2; $var3/$var4” | bc` usando a calculadora BC”*

Nesse trecho a última linha é alterada para redirecionamento para a calculadora BC

Observe que `` redirecionam para uma saida de comando e | redireciona para a calculadora bc

O scale=2 sinifica para a BC que queremos 2 casas decimais e o ; finaliza o final deste comando, seguido dos elementos da operação

Na chamada de funções podemos indicar parametros a serem usados dentro do script

*./parametros.sh 2 3 5 A*

No script pode ser utilizado o parametro informado atravez das variaveis de parametro de $0 até $9

Sendo que $0 é o nome do arquivo e $# é a quantidade de parametros passados

*echo O quadrado de $1 é $[$s1\*$1]*

*echo Foram informados $# parametros para o script #0*

Perceba que se no script for utilizado um uma variavel de posição de parametro que não foi declarado, retornará um erro.

O *shift* rebaixa cada parametro em uma posição, ou seja, desloca cada parametro em sua posição relativa.

*shift* Serve para manipulação de parametros de linha de comando, funciona até que $1 seja descartado

*i=1*

*while [ -n “$1” ]*

*do*

*echo O parametro $i é $1*

*i=$[ $i+1 ]*

*shift*

*done*

O script vai rebaixando o parametro de linha de comando até que $1 fique vazio

i é um contador apenas e o *-n* testa se a variavel está vazia.