Introdução à Ciência da Computação

Shell Script – parte III

Professor Iago Augusto de Carvalho

iago.carvalho@unifal-mg.edu.br

1

Usando a calculadora do Bash - bc

A calculadora do bash é na verdade uma linguagem de programação que permite executar expressões de ponto flutuante no terminal.

Ela reconhece:

Números inteiros e ponto flutuante

Variáveis simples e arrays

Comentários estilo na linguagem C (/\* \*/)

Expressões matemáticas

Declarações condicionais if-then, while

Funções

2

Usando a calculadora do Bash - bc

Para acessar a calculadora do bash no shell digite o comando:

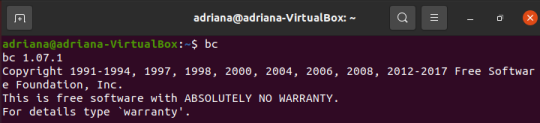
**bc**

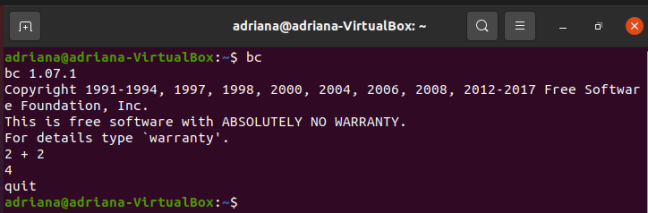
Assim, entrará no modo interativo da calculadora. Digite então as expressões que deseja calcular e pressione Enter.

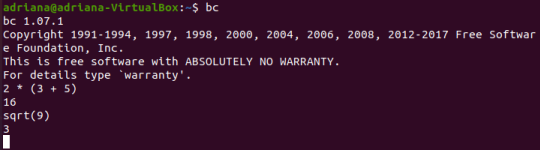
Para sair da calculadora, digite:

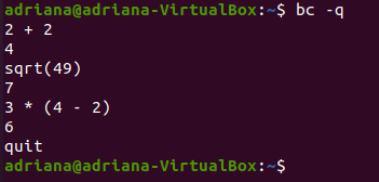
**quit**

3

Usando a calculadora do Bash - bc 

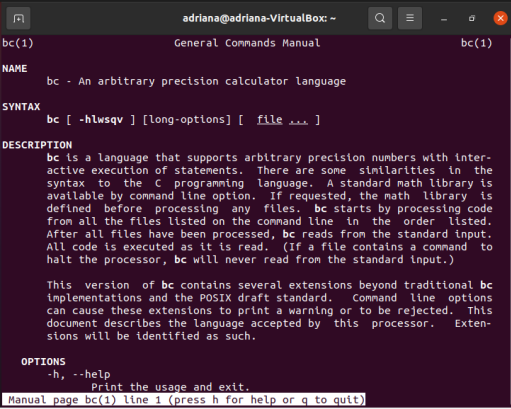
4

Usando a calculadora do Bash - bc 



5

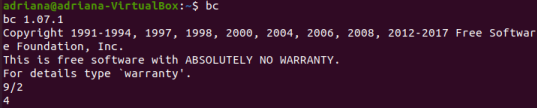
Manual bc

6

Aritmética de Ponto Flutuante na calculadora - bc

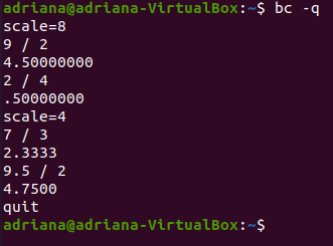
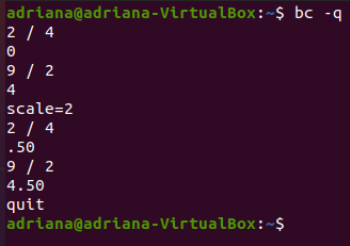
É controlada pela variável especial **scale**. Você configura seu valor para o número desejado de casas decimais que necessita nos resultados.

Por padrão, a variável scale vem definida com o valor zero(0).

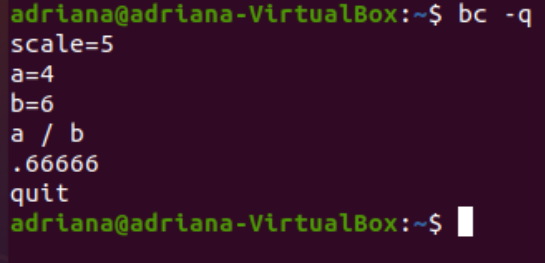


7

Aritmética de Ponto Flutuante na calculadora - bc

8

Aritmética de Ponto Flutuante na calculadora - bc

Cálculo com variáveis 

9

Usando a calculadora bc em scripts

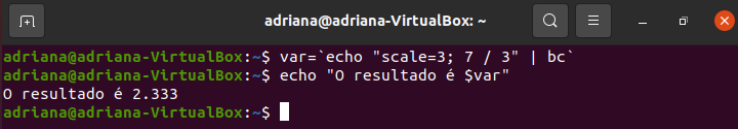
Para usar a bc em um script, use a crase (backtick`) para rodar o comando e atribuir seu valor a uma variável:

**variável=`echo “variáveis; expressão” | bc`**

Exemplo – Digite os comandos a seguir no terminal:

**var=`echo “scale=2; 2 / 5” | bc`**

**echo “Resultado: $var”**

****10

Usando a calculadora bc em scripts

1. Criando script para cálculos simples:

**#!/bin/bash**

**var1=100**

**var2=30**

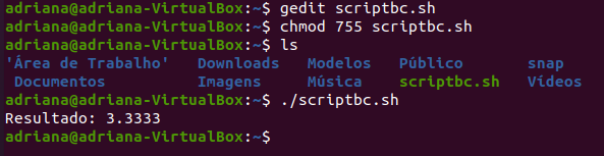
**var3=`echo “scale=4; $var1 / $var2” |bc`**

**echo “Resultado: $var3”**

Essa técnica funciona bem para cálculos simples, mas em cálculos mais complexos é recomendável usar a técnica do redirecionamento de entrada inline.

11

Usando a calculadora bc em scripts 

12

Usando a calculadora bc em scripts

2. Criando script para cálculos mais complexos:

**#!/bin/bash**

**var1=6**

**var2=5**

**var3=4**

**var4=`bc << EXP**

**scale=4**

**a=($var1 \* $var2)**

**b=($var3 + $var1)**

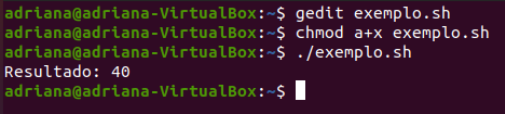
**a+b**

**EXP**

**`**

**echo “Resultado: $var4”**

13

Usando a calculadora bc em scripts 

14

Usando a calculadora bc em scripts 15

Status de Saída de Comandos

Cada comando que roda no shell usa um valor de status de saída para indicar ao shell que o processamento terminou.

O status de saída é um inteiro entre 0 e 255.

A variável especial **$?** armazena o valor do status de saída do último comando executado.

O status de saída de um comando executado com sucesso é 0 (zero). Se houver erro, será um inteiro positivo.

16

Status de Saída de Comandos

| **Código**  0 | **Significado**  Comando completado com sucesso |
| --- | --- |
| 1 | Erro geral desconhecido |
| 126 | O comando não pode ser executado (permissões) |
| 127 | Comando não encontrado |
| 130 | Comando finalizado com Ctrl + C |

Para ver o código de status de um comando, digite **echo $?** logo após o término de sua execução.

17

Status de Saída de Comandos



18

Comando exit

Por padrão, seu shell script finaliza com o status de saída do último comando executado no script. É possível alterar esse comportamento para retornar seu próprio código de status. O comando exit permite especificar um status de saída quando o script finaliza.

Exemplo:

**#!/bin/bash**

**var1=10**

**var2=2**

**var3=$[$var1 \* $var2]**

**echo $var3**

**exit 6** #Retorna o código de saída 6

E também podemos usar variáveis como parâmetro do comando exit:

**exit $var3**

19

Comando exit



20

Comando exit

21

Decisão condicional: If-Then

Sintaxe:

**if comando then**

**comandos fi**

**ou**

**if comando; then comandos**

**fi**

A declaração if do shell bash executa o comando definido na linha if.

Se o status de saída do comando for zero, os comandos listados após a seção then serão executados.

Caso contrário, esses comandos serão ignorados.

22

Decisão condicional: If-Then

Exemplo: 

**#!/bin/bash**

**#Exemplo de condicional simples em um script if cd /**

**then**

**echo “Diretório raiz encontrado!”**

**fi**

****

23

Decisão condicional: If-Then

Exemplo: 

**#!/bin/bash**

**#Exemplo de condicional simples em um script**

**if cd /x**

**then**

**echo “Diretório raiz encontrado!”**

**fi**

****24

Decisão condicional: If-Then-Else

**if comando**

**then**

**comandos**

**else**

**outros comandos**

**fi**

O condicional composto permite executar um bloco de código caso o comando testado retorne código de status zero, e outro bloco de código caso retorne status diferente de zero.

25

Decisão condicional: If-Then-Else

Exemplo: 

**#!/bin/bash**

**#Exemplo de condicional composto em um script**

**if ls /home/adriana/teste**

**then**

**echo “Diretório teste encontrado!”**

**else**

**echo “Diretório teste não encontrado!”**

**fi**

****

26

Decisão condicional: If-Then-Else 



27

Condicional if aninhado

Às vezes é necessário verificar várias situações relacionadas em ao script.

Em vez de ter que escrever declarações if-then separadas, você pode usar uma versão alternativa da seção else, chamada elif.

O **elif** continua uma seção else com outra declaração if-then.

O shell bash executará as declarações if em ordem, e apenas a primeira que retornar status de saída zero terá a seção then correspondente executada.

**if comando1**

**then**

**comandos**

**elif comando2 then**

**comandos 2**

**elif comando3 then**

**comandos 3**

**else**

**comandos-else fi**

28

Condicional if aninhado

29

Condicional if aninhado



30

Comando test

A declaração if-then não consegue avaliar nenhuma condição que não seja o código de status de saída de um comando.

Porém, é possível avaliar outras condições usando o comando test em uma declaração if-then.

O comando **test** avalia uma condição, e se ela retornar true(V), o comando test retorna o código de status de saída igual a zero. Caso contrário, retorna status de saída igual a 1.

31

Uso do test com if-then

**if test condição**

**then**

**comandos**

**fi**

O shell bash fornece uma maneira alternativa de declarar o comando test com if-then:

**if [ condição ]**

**then**

**comandos**

**fi**

Os colchetes definem a condição usada pelo test. Deve haver espaços antes e depois da condição.

32

Comando test – classes de condições

O comando **test**

pode avaliar três

classes de

condições:

Comparações

numéricasComparações de

StringsComparações de

Arquivos

33

Comparações numéricas com test 

Bóson Treinamentos

Avaliam tanto números quanto variáveis.

O comando test não suporta aritmética de ponto flutuante.

34

Comparações numéricas com test

#!/bin/bash

#Comparações numéricas com test e if-then var1=10

var2=15

if [ $var1 -gt 9 ]

then

echo “A variável de valor $var1 é maior que 9” fi

if [ $var1 -eq $var2 ]

then

echo “Os valores são iguais”

else

echo “Os valores são diferentes”

fi

35

Comparações numéricas com test 36

Comparações numéricas com test

37

Comparações de string com test

O comando test também permite realizar comparações entre valores de strings. Bóson Treinamentos

38

Comparações de string com test

#!/bin/bash 

#Testar igualdade de strings

var=“adriana”

if [ $USER = $var ]

then

echo “Olá $var”

fi



39

Comparações de string com test

#!/bin/bash 

#Testar igualdade de strings

var=“icc”

if [ $USER = $var ]

then

echo “Olá $var”

else

echo “$var não é usuário”

fi



40

Comparações de string com test

#!/bin/bash 

#Testando diferença de strings

var=morango

if [ $var != morango ]

then

echo “A fruta não é morango, é $var”

else

echo “A fruta é $var”

fi



41

Comparações de string com test

#!/bin/bash 

#Testando diferença de strings

var=manga

if [ $var != morango ]

then

echo “A fruta não é morango, é $var”

else

echo “A fruta é $var”

fi



42

Comparações de string com test

#!/bin/bash

#Testar se variável possui conteúdo

var=abacaxi

var2=‘’ #variável vazia aspas simples

if [ -n $var ]

thenecho “A variável não está vazia, contém o valor $var” elseecho “A variável está vazia”

fi

if [ -z $var2 ]

thenecho “Variável está vazia”

elseecho “Variável não está vazia”

fi

43

Comparações de string com test

44

Comparações 

de Arquivos

As comparações de

arquivos são o tipo de

comparações mais

poderosas e mais usadas

em shell scripting.

O comando test permite

testar o status de arquivos

e diretórios no sistema de

arquivos Linux.

45 Bóson Treinamentos

Comparações de arquivos

#!/bin/bash 

#Verificar se o diretório HOME do usuário existe # e mostrar seu conteúdo

if [ -d $HOME ]

then

echo “Seu diretório home existe e o conteúdo é:” cd $HOME

ls -l

else

echo “Diretório não encontrado”

fi

46

Comparações 

de arquivos

47

Comparações de arquivos

#!/bin/bash

#Verificar se um objeto é um arquivo

if [ -e $HOME ]

then

echo “O objeto existe. Vamos ver se é arquivo ou diretório” if [ -f $HOME ]

then

echo “É um arquivo”

else

echo “É um diretório”

fi

else

echo “Objeto não encontrado”

fi

48

Comparações de arquivos

49

Comparações de arquivos

#!/bin/bash

#Verificar permissão de leitura em um arquivo

arquivo=/etc/passwd

#testar se o arquivo existe

if [ -f $arquivo]

then

#Existe. Testar se o usuário tem permissão de leitura

if [ -r $arquivo ]

then

echo “Possui permissão de leitura. Mostrando as 5 últimas linhas:” tail -5 $arquivo

else

echo “Sem permissão de leitura”

fi

else

echo “Arquivo não encontrado”

fi

50

Comparações de arquivos

51

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/

52