Programação Shell Script

REDES DE COMPUTADORES

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

O comando echo

* O comando **echo** é utilizado para **mostrar texto** na tela e/ou **valores das variáveis**.

O comando echo

Sintaxe

* echo [opções] [string, variáveis ...]

Opções

* -n Anula a tag de nova linha

Tags

* \a - alerta (barulho de sino)
* \b - backspace
* \n - nova linha
* \t - um tab horizontal
* \v - um tab vertical
* \\ - colocar uma barra em um texto

O comando echo

* Digite o exemplo abaixo:
  * $ echo “ O rato roeu a roupa do rei de \a\t\vroma\n”

* Para ***mais opções***, visualize o ***manual do comando***:
  * man echo

O comando echo

* Como podemos mostrar na tela um *texto colorido* ou em **negrito**?

O comando echo

* Digite o seguinte comando e veja o resultado:
  * $ echo –e “/33]34m Ola Mundo Colorido!”

O comando echo

* $ echo –e “\033[0;31m Ola Mundo Colorido!”

1) \033 - É um ***caractere de escape***. Ele vai passar informações ***para o console*** .
2) [0;30m - Aqui nós estamos ***configurando a cor de fundo*** do console utilizando outro caractere de escape.
3) Só então ***imprime na tela a mensagem*** na cor vermelho...

Perceba que ele irá manter o console com esta cor. Para voltar ao normal abra uma nova sessão.

O comando echo

* Outras cores:

```
corPadrao="\033[0m"
preto="\033[0;30m"
vermelho="\033[0;31m"
verde="\033[0;32m"
marrom="\033[0;33m"
azul="\033[0;34m"
purple="\033[0;35m"
cyan="\033[0;36m"
cinzaClaro="\033[0;37m"
pretoCinza="\033[1;30m"
vermelhoClaro="\033[1;31m"
verdeClaro="\033[1;32m"
amarelo="\033[1;33m"
azulClaro="\033[1;34m"
purpleClaro="\033[1;35m"
cyanClaro="\033[1;36m"
branco="\033[1;37m"
```

Expressões Aritméticas em Shell Script

Sintaxe

* expr operando1 operador-matematico operando2

Exemplos

* $ expr 2 + 4
* $ expr 3 - 2
* $ expr 20 / 4
* $ expr 32 % 3
* $ expr 10 * 4
* $ echo `expr 2 + 4`

Expressões Aritméticas em Shell Script

* Como se chama a operação que utiliza o símbolo % como operador?

Expressões Aritméticas em Shell Script

Mod

* $ expr 32 % 3 - Esta operação é conhecida como mod.
  * Pega o ***resto da divisão***.
  * 32 % 3 = 2 (resto da divisão)

* **ATENÇÃO!** - A multiplicação é feita utilizando * e não somente o *
  * $ expr 10 * 4

Expressões Aritméticas em Shell Script

* Preste atenção ***na ultima linha*** de comando:

$ echo `expr 2 + 4`

1) Após o comando echo utilizamos o ***caractere crase*** (`) e **não** ***aspas simples***.
2) O comando expr sempre termina com o ***caractere crase*** No exemplo o resultado da soma será mostrado na tela, 6
3) Se você tentar utilizar ***aspas simples*** ou ***duplas***, o comando não irá funcionar

Ex:

* $ echo "expr 2 + 4" # ISSO IRA IMPRIMIR expr 2 + 4
* $ echo 'expr 2 + 4' # ISSO IRA IMPRIMIR expr 2 + 4

Exercícios

* Utilizando o ***caractere crase*** e o comando ***expr***, crie um script que apresente o resultado na tela conforme abaixo.

$ 10 + 20 = 30

* Melhore seu script utilizando variáveis (***UDV***) para armazenar os valores dos operandos da soma acima.

Tipos de Aspas

* Como funcionam os diversos tipos de aspas em Shell Script.

Tipos de Aspas

* Em Shell Script existem **3 tipos de aspas**:

Aspas	Nome	Significado
“ ”	Aspas Duplas	As aspas duplas transformam tudo em texto, inclusive comandos, que perdem o seu significado. (Exceto o \ e o $)
‘ ’	Aspas Simples	Tudo que é colocado entre aspas simples não pode mais ser modificado.
` `	Aspas deitadas (crase)	São utilizadas para executar comandos de dentro do seu script

Tipos de Aspas

* Exemplos
* $ echo "A data de hoje eh: date"
  * Vai imprimir ***apenas a mensagem***: A data de hoje eh: date.
  * O comando date **não** ***será interpretado***
* $ echo "A data de hoje eh `date` "
  * Vai imprimir: A data de hoje eh: Seg Fev 11 15:31:50 BRST 2008.
  * O comando date ***será interpretado***
* :

Exercício

* Como verificar se o seu script **executou corretamente**?

<< Ponto Extra >>

Resposta

* Basta verificar o *Status de Saída*

* O que é o Status de Saída de um Script?

Status de Saída do Script

* No Linux, por padrão, quando um comando ou um script shell é executado, ***o console retorna dois tipos de valores***
  * Se o valor retornado for o ***zero (0),*** o comando foi executado com ***sucesso***
  * Se o valor retornado ***não for zero***, o comando **não** foi executado com sucesso. (***Provável erro***)
* Esses valores são conhecidos como ***status de saída*** ou **Exit Status** e podem ser utilizados para verificar se o comando ou o script ***executaram com sucesso*** ou ***não***

Status de Saída do Script

* Como podemos *descobrir* o *status de saída* do **nosso** script shell?

Status de Saída do Script

Simples.

* Para descobrir o status de saída basta utilizar a ***variável especial*** do shell apresentada abaixo:

$?

Verificando o Status de Saída do seu Script

* Digite o seguinte comando para ***remover um arquivo*** que **não** existe no seu home de usuário
  * rm arqaluno
* Ele irá mostrar o seguinte erro:
  * rm: cannot remove `unkowm1file': No such file or directory
* Digite em seguida o comando para verificar o ***Status de Saída***:
  * $ echo $?

Verificando o Status de Saída do seu Script

* O resultado irá mostrar na tela um ***valor diferente de zero*** para indicar ***erro na execução*** do comando anterior

* Tente agora:
  * $ ls
  * $ echo $?

* O resultado irá mostrar na tela o ***valor zero (0),*** que indica que o seu ***comando foi executado com sucesso***

Exercício

* Teste os comandos abaixo e observe o ***status de saída*** de cada um deles:

1)
* $ expr 1 + 3
* $ echo $?

2)
* $ echo BemVindo
* $ echo $?

3)
* $ isso funciona?
* $ echo $?

4)
* $ date
* $ echo $?

5)
* $ echon $?
* $ echo $?

Variáveis Estáticas x Dinâmicas

* Até agora apenas utilizamos *variáveis estáticas*...
* Ou seja, os valores foram definidos dentro do script, *diretamente na variável.*

Exemplos

* x = 10
* *y = 20*
* *nome = aluno*

Variáveis Dinâmicas

* O que são **variáveis dinâmicas**?

Variáveis Dinâmicas

* São aquelas cujo os valores são ***definidos pelo usuário*** em ***tempo de execução do Script***.

* Ou seja, o usuário digita o valor que ele quiser durante a execução do script.

* Como podemos ler dados **digitados pelo usuário** durante a execução do script?

O comando read

* Comando **read**

* O comando **read** é utilizado para adquirir dados da entrada padrão do usuário (teclado) e armazena-los em uma variável.

Sintaxe

* read variavel1, variavel2, ..., variavelN

O comando read - exemplo

* O script a seguir primeiro solicita ao usuário seu ***nome*** e espera que ele o digite ***através do teclado.***

* Depois de digitar o nome, o usuário é obrigado a ***digitar a tecla*** "**ENTER**"

* O valor ***será armazenado*** em uma variável chamada ***fnome***

O comando read - exemplo

```
#!/bin/bash
#Script para ler um nome do teclado
#Nome: exemploread.sh
echo "Digite seu primeiro nome:"
read fnome
echo "Fala $fnome aluno!"
```

* Relembrando:
* Para ***executar*** o script:
  * $ chmod 755 exemploread.sh
  * $ ./exemploread.sh

Utilidade Linux

* Utilidade Linux
  * Trabalhando com arquivos utilizando caracteres **coringa.**

Coringas	Significado	Exemplos
*	Utilizado para encontrar *qualquer string* ou *grupo de caracteres*	•$ ls * - Mostra todos os arquivos •$ ls a* - Mostra todos os arquivos que começam com a letra a •$ ls *.c - Mostra todos os arquivos que possuem a extensão .c •$ ls ut*.c - Mostra todos os arquivos que possuem a extensão .c e que começam com as letras ut
?	Utilizado para encontrar *um único caractere*	•$ ls ? - Mostra todos os arquivos que possuem apenas 1 caractere •$ ls ?? - Mostra todos os arquivos que possuem 2 caracteres •$ ls ??? - Mostra todos os arquivos que possuem 3 caracteres •$ ls fo? - Mostra todos os arquivos que possuem 3 caracteres e que começam com as letras fo
[...]	Todos os arquivos que iniciem com *qualquer uma das letras* entre colchetes	$ ls /bin/[a-c]* - Mostrar todos os arquivos que começam com as letras a, b ou c

Obrigado.

Contato: joaopauloaramuni@gmail.com

UNIVERSIDADE FUMEC

50 Anos