Shell Script do zero

Aula 9 – Operações matemáticas e inicialização de scripts

Na aula passada já vimos um pouco de matemática quando somamos o valor da variável +1 para contarmos o loop:

```
VOLTA=$[ $VOLTA + 1 ]
```

Ou usando o sinal de menos ↓↓↓ (esquece multiplicação e divisão nesta sintaxe)

```
VOLTA=$[ $VOLTA - 1 ]
```

Quando o script precisar literalmente fazer contas igual uma calculadora, usamos a sintaxe abaixo, deixando a conta sempre entre os dois parênteses associando o resultado a uma variável, neste comando não faz diferença escrever com ou sem espaço.

```
#!/bin/bash

CONTA=$((10 * 2))

echo $CONTA
```

Os símbolos matemáticos usados no shell são os que estamos acostumados

\+

\-

*

/

Acima a variável CONTA é igual ao resultado da operação entre parênteses e no echo eu mostro o resultado, eu poderia por exemplo usar este resultado em algum comando do script.

Uma vez usei este conceito para redundância, se o comando retornasse 64 eu executaria o comando 2, só que apenas uma verificação poderia ter falhas, então coloquei três verificações somei todas elas usando suas variáveis e na lógica de comparação eu dizia que poderia executar o comando **se** o valor fosse igual ou maior que 64, ou seja, se até duas verificações falhassem eu conseguiria o valor 64 o que indicava que pelo menos uma estava funcionando. Abaixo uma aplicação das operações matemáticas usando variáveis.

```
#!/bin/bash

N1=$"3"
N2=$"2"

CONTA=$(($N1 * $N2))
```

Infelizmente o shell só mostra resultados inteiros se a resposta for fracionária ele mostra como inteiro ou as vezes não funciona.

É difícil dar exemplos práticos, eu usei pouquíssimas vezes contas em script, a não ser que o seu programa seja mais voltado para matemática.

Devemos salientar que se o valor for considerado matemático devemos ter cuidado com as aspas no if, while etc.

Aqui fazemos o comando considerando a condição como um valor matemático	E aqui como um texto, colocando aspas na variável e item a ser comparado
`if [ $CONTA -eq 1 ];then`	`if [ "$CONTA" = "1" ];then`

Podemos ter outras variações que podem dar certo, você sabendo estas 2, vamos complicar pra quê.

Inicializando scripts com o sistema e programando-os

Daqui para baixo não se preocupe em decorar nada, use para consultas futuras.

Além das opções abaixo você pode também colocar o script nos Aplicativos de Sessão

Inicialização

Rc2.d

Com o diretório rc2.d o script será uma das últimas coisas a serem executadas como root, basta seguir o passo a passo abaixo:

coloque o script em → /etc/init.d
coloque seu atalho em → /etc/rc2.d/

Atalhos começando com nome S99 são os últimos executados, podendo ficar assim:
/etc/rc2.d/S99meus_cript

Se você quiser trabalhar melhor a ordem de execução, é só dar um "ls /etc/rc2.d" e verificar os scripts que constam lá.

Rc.local

Basta colocar o endereço do script no arquivo-texto → etc/rc.local

exemplo:
/etc/meu_script

Inicializando com usuário especifico:
sudo -u usuário /etc/script

Também podemos programar o script para executar em determinados dias e horários.

Sempre deixe a ultima linha do crontab vazia

Cron

/etc/crontab
service cron restart

Editamos o arquivo /etc/crontab levando em consideração os campos abaixo.

Campo	Função	Preenchimento
1º	Minuto	0-59
2º	Hora	0-23
3º	Dia do Mês	1-31
4º	Mês	1-12
5º	Dia da Semana	0 Domingo, 1 Segunda ...
6º	usuário	root luiz etc.
7º	Programa para execução	Comando

*O 6º campo pode ser omitido, mais evite de fazer isto, porque costuma dar pau

Ficando assim → 20 10 2 12 2 luiz /home/luiz/meu_script

Do comando gerado acima podemos entender → aos 20 minutos das 10 horas do dia 2 de dezembro numa terça feira o usuário luiz executará o "meu_script" que está em /home/luiz

Referência do Cron: http://www.hardware.com.br/dicas/cron.html