

Shell Script do zero

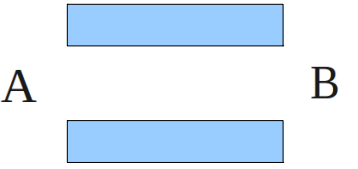
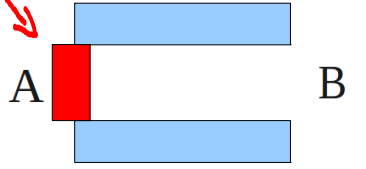
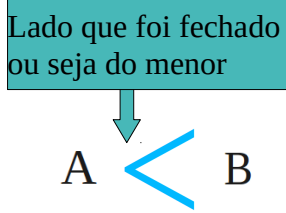
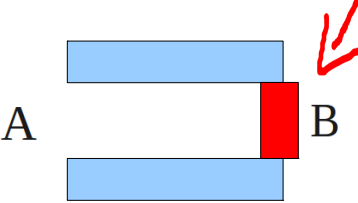


Aula 7 – Fazendo loops com o “for” e script remoto

Você já parou para pensar como podemos deixar um script independente, que fique ali de prontidão vigiando as situações e executando comandos de acordo com o que acontece, sem precisar de nenhum gatilho humano, o tempo que for, como se fosse uma sentinela.

Isto é possível com o “for”, ele é um dos comandos que possibilita a inserção de loops em scripts, a sua tradução é “para”, ou seja, para determinada condição faça o comando enquanto ela for verdadeira.

Maior e menor

Para usar o comando “for” temos que saber o sentido destes dois comparadores (“>” “<”), eu uso a seguinte regra para não confundi-los:

| | | |
|--|--|--|
| <p>Pense em duas barras</p>  | <p>Eu quero dizer para o shell que A é menor que B então eu diminuo o lado da barra em que está o A.</p>  | <p>Ficando assim:</p>  |
| <p>Agora eu quero dizer que B é menor que A (ou A é maior que B). Então é só inverter a lógica.</p>  | <p>Ficando assim:</p>  | <p>Eu posso trocar o B de lugar com o A? Ficaria a mesma coisa!</p>  <p>O comando que apresentarei não podemos inverter os lados.</p> |

Entendendo o comando e criando loops limitados

Sintaxe

Os parênteses estarão **sempre** desta forma

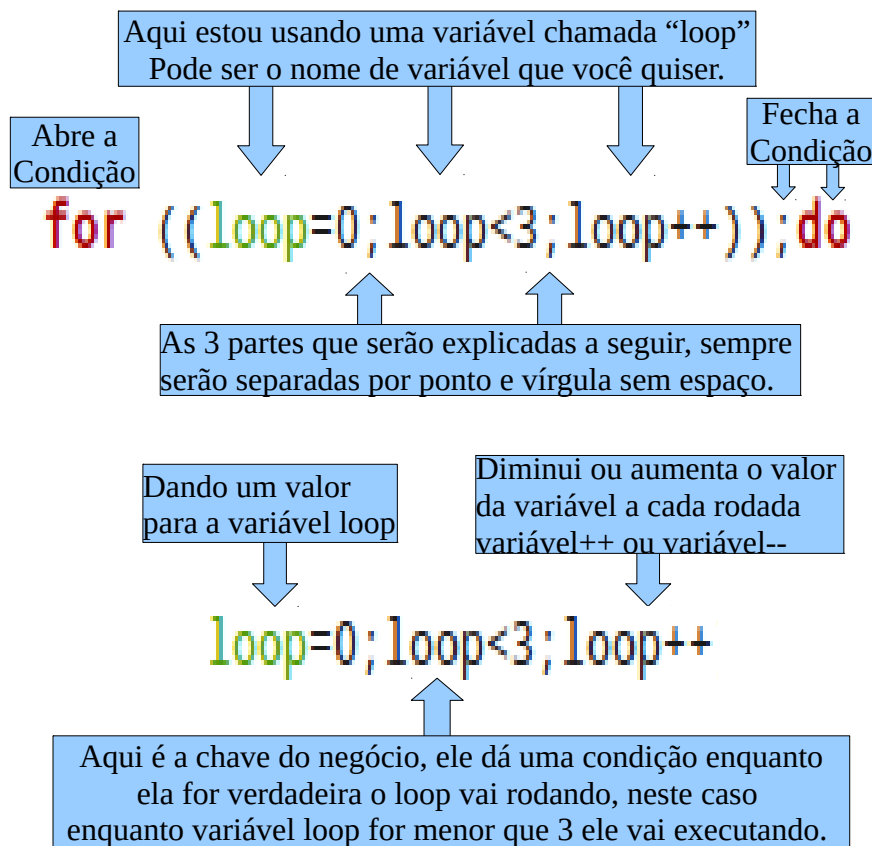
```
for ((loop=0;loop<3;loop++));do
```

Os códigos que ficarão rodando no loop ficam aqui dentro

```
done
```

Aqui fechamos o "for" com o "done"

Continuando com a sintaxe



A tradução do comando "for":

```
for ((1loop=0;2loop<3;3loop++));do
```

- 1 Para a variável loop que vale zero
- 2 Rode os comandos enquanto ela for menor que 3
- 3 E a cada rodada acrescente + 1.

Atenção

Este loop rodará 3 vezes, nestes valores da variável loop → → → 0, 1 e 2

Acontece isto:

Ele verifica a condição e roda uma vez, \$loop passará a valer 1, ele verifica se loop estiver menor que 3 ele roda novamente, depois ele verifica e vê que loop vale 2 e que a condição ainda é verdadeira e roda novamente, agora loop vale 3 e não é menor do que 3, então ele deixa de executar o "for" e continua o script, já que a condição passou a ser falsa.

Como enchemos o comando de placas, coloquei o código ao lado sem nenhuma poluição. → → →

É interessante você colocar o comando: `echo $loop` (dentro do “for”) para ir vendo na tela quantos loops dará

```
#!/bin/bash
for ((loop=0;loop<3;loop++));do
    echo $loop
done
```

A condição deve ser verdadeira ao passar pelo “for” na primeira vez, caso seja falsa passará direto e não voltará para executá-lo, então preste atenção na hora de montar a lógica. Salvo em momentos que você colocar um loop dentro de outro loop e o primeiro rodará até criar condições de cair no segundo.

Criando loops infinitos

Trazendo a lógica que vimos acima, para criarmos loops infinitos, basta fazer uma condição que jamais deixará de ser verdadeira. Exemplo:

```
for ((loop=2;loop>1;loop++));do
```

Conforme mostrado acima, você deve concordar comigo que:

- Loop tem o valor de 2
- Para rodar o “for” loop deve ser maior que 1 (e a condição já é verdadeira)
- E o valor de loop nunca vai diminuir, sempre vai aumentar ++

Ou seja, o loop nunca será menor que 1 para que a condição seja falsa, então ele rodará “para sempre”.

Se criarmos um loop com comandos rápidos, o script dará milhares de voltas em poucos segundos, é recomendado colocar um `sleep` dentro do “for” limitando assim o tempo das rodadas.

Tenho mais duas formas de escrever o “for” que funciona um pouco diferente do que expliquei anteriormente, como eu nunca usei na forma prática, vou apenas apresentar.

```
for cor in azul vermelho amarelo verde
do
    echo $cor
done
```

Aqui ele rodará 4 vezes, uma para cada parâmetro, sendo que cada vez que ele rodar, a variável “cor” terá o valor de um parâmetro, que são: azul, vermelho ...

```
for PAR in $*
do
    echo $PAR
done
```

Usamos o mesmo conceito neste “for”, a diferença é que o usuário digita os parâmetros (quantos quiser). Ele digita o comando do script, espaço, depois os parâmetros. `./script parâmetro1 parâmetro2 parâmetro3 ...`

Veremos o conceito de parâmetros futuramente, ele tem utilidade no script como um todo e não somente no “for”.

Imagine que você está saindo de casa faltando 20 minutos para terminar aquele download e você precisa ficar presente estes 20 minutos para depois abrir outro programa de download, torrent por exemplo já que ele atrapalharia o download atual, depois disto seria necessário desligar o pc após 2 horas, converter aquele vídeo pesado ou dar um comando demorado etc. Mas como fazer isto sem estar presente?

Seja bem-vindo ao nosso exercício 4 – script remoto

O script remoto ficará ao seu dispor para executar qualquer comando, usaremos o Dropbox, assim pelo celular ou qualquer dispositivo com internet você poderá comandar o script, mas fique a vontade de não usar o Dropbox e colocar os arquivos em qualquer diretório do pc, já que a nossa intenção é o aprendizado e testar nosso script.

Para o script apontar corretamente a home de qualquer usuário que o execute, basta usar uma das duas opções abaixo:

`$HOME` ou `/home/$USER`

Exemplo: `echo "teste" > $HOME/arquivo`
se transformará em: `/home/luiz` por exemplo

Funcionamento

- O script rodará 1 vez a cada 1 minuto
- A cada rodada ele lê o arquivo-texto
- No arquivo-texto escreveremos o comando desejado
- Ele executa o comando digitado e limpa o arquivo-texto

Avisos

- Zere ou crie o arquivo “comando” usando o → `echo "" > $HOME/comando` (fora do loop)
- Você deverá usar um certo recurso para o script não ficar agarrado no comando (aula 1)
- Use → `echo $loop` dentro do “for”, senão vai parecer que deu pau
- Na construção do script diminua o tempo de loop para fazer os testes

Se o script for desligar o computador ou qualquer outro comando que exija privilégios, rode ele como root, caso contrário execute como usuário normal que é bem melhor já que abrindo um programa como root as configurações estarão diferentes do normal.

Alguma dúvida de como o script funciona?
Veja o vídeo com ele rodando, no link abaixo

http://www.mediafire.com/download/7ceth5ukcc4c5vq/video_exercicio_4.ogv

Incremento

- Se escrever → vivo? - ele responde → “sim vivo, esperando o comando”, juntamente com a variável de loop para diferenciar as escritas. A resposta estará num arquivo de leitura (avisos).

O exercício está no <http://www.mediafire.com/download/lah1i112l4qae9b/remoto> com ele podemos abrir por exemplo o torrent → `vuze` e quando quiser finalizá-lo → `killall vuze`.

Até a próxima, onde aprenderemos sobre os comandos while e until