01

Consumo de memória do sistema

Transcrição

Nesta aula, verificaremos o consumo de memória do sistema, a pedido de nossos diretores da *Mutillidae*. Foi comentado que o consumo de memória estaria comprometendo a performance dos serviços oferecidos pela *Mutillidae*. A nossa missão é **monitorar o consumo de memória do sistema**. Caso ele atinja um nível crítico, devemos passar um e-mail para o administrador de sistemas, para que assim ele possa ver o que está acontecendo.

Para que possamos ver os valores de memória utilizada, e a memória do sistema, podemos utilizar o comando `free`. Informações sobre a RAM do sistema e a memória SWAP, são mostradas na tela.

> Memória SWAP é a memória utilizada quando a RAM está lotada.

Filtraremos somente a linha da memória RAM. E para essa tarefa, utilizaremos o **`grep`**, como já fizemos em etapas anteriores. A ideia é redirecionar a saída do comando `free` através do *pipe* para o `grep`!

```
$ free | grep
```

De acordo com o comando acima, o grep fará o filtro somente na linha especificada. Poderíamos colocar exatamente igual ao campo de memória: o **Mem**, ou seja, com a primeira letra em maiúsculo, e as outras duas em minúsculo.

Mas para não nos preocupar em deixar que o grep faça esse filtro para nós, baseado em letras maiúsculas ou minúsculas, podemos colocar o comando `-i` de **insensitivo**. Assim a pesquisa não será feita de acordo com o tamanho das letras e sim somente com a palavra especificada como parâmetro.

```
$ free | grep -i mem
```

O resultado será parecido com esse:

```
              total        used        free      shared  buff/cache   available
Mem:        2047912     1259960      255676       17220      532276      591832
```

Após ter filtrado os resultados relativos à memória RAM do sistema, precisamos da *memória usada* dividida pela *memória total*, para que tenhamos a relação entre elas. Caso elas estejam acima do limite, mandaremos um e-mail para o administrador de sistemas.

Dessa forma, precisamos pegar os campos dois e três respectivos ao **total** e a **memória utilizada**. Como já fizemos também, utilizaremos o `awk` para realizar essa tarefa.

Para poder pegar o resultado da segunda coluna, o awk imprimirá o `$2`. Para a terceira coluna, o awk imprimirá o resultado do `$3`. Ao executar o código a seguir, o que ocorrerá?

```
$ free | grep -i mem | awk '{ print $2 }'
```

Isso mesmo! Temos exatamente o valor que nos interessa para realizar essa conta entre a memória atual consumida e a memória total do sistema. Já que esse comando funciona, vamos copiá-lo para começar a montar o script.

Como já estamos no diretório de Scripts (caso você ainda não esteja, acesse-a através do comando `cd /Scripts`), criaremos um novo script chamado de `verificacao-memoria-consumida.sh`.

```
$ nano verificacao-memoria-consumida.sh
```

A primeira linha do script deve ser o interpretador. Logo após o interpretador, podemos colar o comando que testamos no terminal:

```
#!/bin/bash

free | grep -i mem | awk '{ print $2 }'
```

Lembrando que, como é um **comando**, temos que envolvê-lo pelo `$()`, e vamos armazenar o resultado desse comando em uma variável:

```
#!/bin/bash

memoria_total=$(free | grep -i mem | awk '{ print $2 }')
```

Podemos utilizar a mesma estratégia para pegar o resultado respectivo à *memória atual consumida*, com a única diferença que o campo respectivo à memória consumida está na **terceira coluna**. Por isso, pegaremos o conteúdo da variável `$3`. O código ficará assim:

```
#!/bin/bash

memoria_total=$(free | grep -i mem | awk '{ print $2 }')
memoria_consumida=$(free | grep -i mem | awk '{ print $3 }')
```

Agora, precisamos ver a forma de dividir a memória consumida pela memória total para poder obter a relação entre elas. Vamos sair do script utilizando o comando "Ctrl + X" e "Y", e voltar para o terminal.

Para realizar essa divisão, precisamos utilizar o comando **`bc`**, passando os valores da divisão como parâmetros: (memória utilizada/memória total).

```
$ bc <<< "scale=2;1260952/2047912"
```

Utilizamos o `scale=2` para ter uma precisão maior, com 2 casas decimais.

O resultado que teremos é `.61`!

Mas, o que acontece? A nossa estratégia é pegar o valor entre a relação da memória consumida com a memória total, e compará-la com o valor limiar que será passado pelos diretores da *Mutillidae*. Entretanto, precisamos comparar esses valores. A primeira coisa que nos vem a mente é utilizar o `if`, correto? Mas, por padrão, o `if` só aceita valores inteiros!

Então, precisamos encontrar outra maneira de resolver isso. Uma delas é **transformar o resutado multiplicando-o por 100**, assim teremos o valor em porcentagem.

```
$ bc <<< "scale=2;1260952/2047912 *100"
```

Agora o resultado é `61.00`. Só que ainda tem o `.00` que não é muito interessante, pois só precisamos do número inteiro. Para remover essa parte que não é interessante para nós, utilizaremos o **awk**.

Vamos redirecionar a saída `61.00` para o awk, onde ele vai **cortar** esse valor justamente no **.** (ponto), entre o valor `61` e `00`.

Após a divisão no ponto, passamos a ter dois campos: o campo **um** `61` e o campo **dois** `00`. Para nós será interessante o campo **um**. Depois que o awk fez o corte, ele terá que mostrar o campo um!

```
$ bc <<< "scale=2;1260952/2047912 *100" | awk -F. '{ print $1 }'
```

Após a execução desse comando, "**61**" foi nos retornado sem ter nenhum ponto flutuante. Se funcionou no terminal, também irá funcionar no script. Faremos a relação entre a memória consumida e a memória total, para ter o valor inteiro que utilizaremos no `if`.

Vamos voltar ao nosso script e colar o comando que acabamos de testar! Para dizer que é um comando vamos envolvê-lo com $(), e armazenaremos o resultado em uma variável:

```
#!/bin/bash

memoria_total=$(free | grep -i mem | awk '{ print $2 }')
memoria_consumida=$(free | grep -i mem | awk '{ print $3 }')
relacao_memoria_atual_total=$(bc <<< "scale=2;$memoria_consumida/$memoria_total *100" | awk -F.
```

Hora de testar o script!

Colocaremos o comando `echo` para imprimir o resultado da variável `relacao_memoria_atual_total`:

```
#!/bin/bash

memoria_total=$(free | grep -i mem | awk '{ print $2 }')
memoria_consumida=$(free | grep -i mem | awk '{ print $3 }')
relacao_memoria_atual_total=$(bc <<< "scale=2;$memoria_consumida/$memoria_total *100" | awk -F.

echo $relacao_memoria_atual_total
```

Com "Ctrl + X" podemos sair, e "Y" para confirmar as alterações.

Vamos rodar o script:

```
$ bash verificacao_memoria_consumida.sh
```

Legal! O script imprimiu o resultado da relação entre as memórias, consumida e total, sendo ela de 61%.

Agora que já temos essa informação, temos que comparar esse valor com o valor limiar passado pelos diretores da *Mutillidae,* e passar o e-mail para o administrador de sistemas.