01

Listando os processos do sistema

Transcrição

Em um certo dia de reunião, os diretores da *Multillidae* reclamaram sobre a existência de alguns processos com grande quantidade de memória alocada e que estariam prejudicando os serviços que a empresa oferece.

Nossa missão é localizar os **dez** processos que estão com uma quantidade maior de memória alocada. Devemos salvar cada um dos processos em um arquivo com o respectivo **nome do processo** e extensão `.log`.

Por exemplo, vamos considerar o navegador **Mozilla Firefox** sendo um destes processos. Encontramos um arquivo chamado `Firefox.log` e dentro dele devem constar as seguintes informações:

- Data
- Horário
- Alocação em Megabytes

Estas informações precisam de uma formatação específica. A **data** deve conter primeiro o `ano`, logo após o `mês` e em seguida o `dia`. Separado por **vírgula**, temos o **horário**, que deve conter primeiro a `hora`, em seguida o `minuto` e por fim o `segundo`. Por último, vem a configuração em MB da alocação de memória para esse respectivo processo.

```
2017-07-21, 15:09:30, 150 MB
```

Vamos localizar esses dez processos, salvá-los com seu nome e sua extensão `.log` e cada um desses arquivos deve conter essas informações.

Testaremos essas saídas no terminal antes de montar o script.

O primeiro passo da estratégia é **buscar os dez processos**. Podemos listar todos os que existem no sistema com o comando `ps -e`. Obtemos um resultado parecido com esse:

```
  PID  TTY          TIME CMD
    1  ?        00:00:04 systemd
    2  ?        00:00:00 kthreadd
    3  ?        00:00:03 ksoftirqd/0
    5  ?        00:00:00 kworker/0:0H
    7  ?        00:00:04 rcu_sched
    8  ?        00:00:00 rcu_bh
    9  ?        00:00:00 migration/0
   10  ?        00:00:00 lru-add-drain
   11  ?        00:00:00 watchdog/0
   12  ?        00:00:00 cpuhp/0
   13  ?        00:00:00 kdevtmpfs
   14  ?        00:00:00 netns
   15  ?        00:00:00 khungtaskd
    ...
```

Na primeira coluna consta o número de **identificação do processo**. Com ele, seremos capazes de filtrar todas as informações que são necessárias para, como o *nome do processo* e o *tamanho de memória alocada*.

Quando executamos o comando `ps -e` , podemos especificar dizendo quais informações de **saída** são respectivas ao número do processo.

```
$ ps -e -o pid
```

Com esse comando listamos somente o `PID` no *output*, que são os números de identificação do processo. Não necessariamente, os processos que estão listados em ordem são de fato os processos que tem uma quantidade maior de memória alocada. É preciso rearranjar os processos para que sejam mostrados primeiro os de maior quantidade de memória utilizada.

Pelo fato de precisarmos fazer uma nova **ordenação**, precisamos pedir para o comando `ps` .

```
$ ps -e -o pid --sort -size
```

Quando teclamos o "Enter", serão listados os processos com maior quantidade de memória alocada. Fizemos uma ordenação dos processos, baseando-se no parâmetro `-size` , que é justamente o tamanho de memória alocada para cada processo.

Observe que não precisamos de todas as informações, e sim, somente dos **dez** primeiros processos dessa lista. Vamos redirecionar a saída do comando que testamos para o **head** que foi visto no curso de *Linux*.

O `head` filtrará automaticamente os dez primeiros processos. Mas para isso, é preciso que o head traga **onze** linhas, pois a primeira será para o cabeçalho e as outras dez serão para os processos.

Vamos redirecionar novamente:

```
$ ps -e -o pid --sort -size | head -n 11
```

Esse foi o resultado que tivemos.

```
  PID
14009
27362
 1445
 1584
13107
 1594
 1633
 1501
  826
 1376
```

Para melhorar o resultado, seria interessante eliminar a linha que contém o `PID` , assim teremos somente as linhas que serão utilizadas. Faremos isso redirecionando toda a saída para o comando `grep` assim ele pegará somente as linhas que contém números. Como podemos fazer isso?

```
$ ps -e -o pid --sort -size | head -n 11 | grep [0-9]
```

Foram filtradas somente as linhas que contém números! Então, com o número de identificação do processo, conseguimos ter várias informações, inclusive com relação ao nome do processo, ao tamanho de memória alocada do processo, etc.

Primeiro, veremos se conseguimos ter o nome dos processos, pois precisamos salvar os arquivos com o nome + `.log` . Faremos esse teste com o processo **14009** para obter seu nome.

```
$ ps -p 14009 -o comm=
```

E temos como resultado o nome do processo: `firefox` !

Vamos para o diretório `/Scripts` , para montar o script que atenderá a requisição dos diretores da *Multillidae*.

```
$ cd Scripts/
$ nano processos-memoria.sh
```

Como já sabemos, a primeira linha deve conter o interpretador e depois vamos trazer os dois comandos que testamos no terminal:

Se são comandos temos que envolvê-los entre `$()` e armazenar o resultado do comando em uma variável.

```
#!/bin/bash

processos=$(ps -e -o pid --sort -size | head -n 11 | grep [0-9])

$(ps -p 14009 -o comm=)
```

Certo, assim, teremos os números dos dez processos, mas para não ficar repetindo o código várias vezes utilizaremos o laço de repetição strong>for para pegar o número de identificação `PID` para que tenhamos as informações de cada processo.

```
#!/bin/bash

processos=$(ps -e -o pid --sort -size | head -n 11 | grep [0-9])
for pid in $processos
do
    $(ps -p $pid -o comm=)
```

Vamos pegar o conteúdo da variável `pid` e imprimimos o valor.

```
#!/bin/bash

processos=$(ps -e -o pid --sort -size | head -n 11 | grep [0-9])
for pid in $processos
do
```

```
    echo $(ps -p $pid -o comm=)
done
```

Para salvar, usamos "Ctrl + X" e "Y".

Vamos verificar se, de fato, ele é capaz de imprimir os nomes desses dez processos. Vamos rodar o script com `bash processos-memoria.sh` .

O resultado será este:

```
firefox
mysqld
compiz
gnome-software
soffice.bin
nautilus
fwupd
indicator-datet
Xorg
hud-service
```

Temos a impressão dos dez processos que estão com maior quantidade de memória alocada! A primeira parte da nossa missão foi concluída. Agora, precisamos nos preocupar com o conteúdo interno dos arquivos: a data, a hora, e a alocação em megabytes.