01

Criando arquivos e salvando data e hora

Transcrição

Até agora o script criado é capaz de descobrir os nomes dos 10 processos com maior quantidade de alocação de memória. Entretanto, daremos foco a parte interna de cada um dos arquivos.

Descobrimos um comando que nos ajudará nessa tarefa: `date` .

O comando `date` nos mostra o dia e o horário de execução. Mas, se lembrarmos da requisição dos diretores da *Multillidae*, eles pediram que a data estivesse formatada da seguinte maneira:

```
ano-mês-dia, horas:minutos:segundos
```

Precisamos formatar a saída para atender ao formato solicitado pelos diretores. Focaremos a nossa atenção na primeira parte da formatação, que se refere à **data**.

Vamos utilizar o comando `$F` que significa **data completa** em inglês.

```
$ date +%F
```

O resultado é o seguinte:

```
2027-07-21
```

Podemos ir mais além desse resultado, concatenando-o com as horas, minutos e segundos.

```
$ date +%F,%H:%M:%S
```

E temos o seguinte:

```
2027-07-21,15:31:58
```

Então, com esse comando, conseguimos resolver esse problema. Vamos copiá-lo para colar no script `processos-memoria.sh` .

```
do
    echo $(ps -p $pid -o comm=)
    date +%F,%H:%M:%S
done
```

Para descobrir o nome do processo, vamos utilizar o comando anterior e salvamos o resultado do comando em uma variável.

```
do
    nome_processo=$(ps -p $pid -o comm=)
    echo $(date +%F,%H:%M:%S)
done
```

Para imprimir *utilizamos o* `echo` e para redirecionar a saída para um arquivo acrescentamos o `>` e o nome_do_arquivo.log

```
echo $(date +%F,%H:%M:%S) > $nome_processo.log
```

Só que sempre que o script for executado, a informação que estiver presente no arquivo `nome_do_processo.log` , vai ser sobrescrita. E não queremos isso! Nosso desejo, na verdade, é ter um histórico das informações.

É importante juntar a nova informação que estamos executando para o arquivo do `nome_do_processo.log` . Para resolver esse problema, basta acrescentar mais um **>>**, assim ele não irá sobrescrever.

Já conseguimos ter a data e o horário, mas, acontece que ainda temos que colocar a memória em megabytes.

Se deixarmos o comando desta forma, o `echo` vai imprimir na mesma linha a data e a hora, só que por padrão, quando ele termina de ser executado, ele vai para a próxima linha. Não queremos isso, pois ainda precisamos colocar a locação em MB após o campo do horário. Para isso, precisamos falar para o `echo` **não** fazer uma quebra de linha.

```
echo -n $(date +%F,%H:%M:%S) >> $nome_processo.log
```

Vamos aproveitar para alterar a formatação para *vírgulas*, para depois poder acomodar os megabytes da memória alocada.

```
echo -n $(date +%F,%H:%M:%S,) >> $nome_processo.log
```

Vamos salvar as alterações com "Ctrl + X" e "Y".

Assim como descobrimos o nome do processo pelo comando, também podemos descobrir o tamanho da memória alocada por ele. É uma forma bem parecida ao que já fizemos. Queremos listar o processo que tenha o número de identificação do PID, imprimindo o tamanho no *output*.

```
$ ps -p 14009 -o size
```

O que foi retornado?

```
 SIZE
947732
```

É mostrado o tamanho da memória alocada para o processo "14009".

Mas, repare que interessante! O número "947732" está em **kilobytes** ,e a requisição dos nossos diretores é de que a informação esteja em **megabytes**. Vamos ver como podemos solucionar esse problema.