Processos no Linux

Introdução

Nos sistemas operacionais, um processo é a forma de representar um programa em execução. É o processo que utiliza os recursos do computador - processador, memória, etc - para a realização das tarefas para as quais a máquina é destinada. Este artigo mostrará os principais conceitos relacionados a processos no Linux e as ferramentas usadas para manipulá-los e gerenciá-los.

Composição de um processo

O sistema operacional lida com uma infinidade de processos e, por isso, é necessário ter meios que permitam controlá-los. Para isso, os processos contam com um conjunto de características, dentre as quais:

- Proprietário do processo;

- Estado do processo (em espera, em execução, etc);

- Prioridade de execução;

- Recursos de memória.

O trabalho de gerenciamento de processos precisa contar com as informações acima e com outras de igual importância para que as tarefas sejam executadas da maneira mais eficiente. Um dos meios usados para isso é atribuir a cada processo um PID.

PID e PPID

Um PID (**P**rocess **Id**entifier) é um número de identificação que o sistema dá a cada processo. Para cada novo processo, um novo número deve ser atribuído, ou seja, não se pode ter um único PID para dois ou mais processos ao mesmo tempo.

Os sistemas baseados em Unix precisam que um processo já existente se duplique para que a cópia possa ser atribuída a uma tarefa nova. Quando isso ocorre, o processo "copiado" recebe o nome de "processo pai", enquanto que o novo é denominado "processo filho". É nesse ponto que o PPID (**P**arent **P**rocess **Id**entifier) passa a ser usado: o PPID de um processo nada mais é do que o PID de seu processo pai.

UID e GID

Conforme já mencionado, cada processo precisa de um proprietário, um usuário que seja considerado seu dono. A partir daí, o sistema saberá, através das permissões fornecidas pelo proprietário, quem pode e quem não pode executar o processo em questão. Para lidar com os donos, o sistema usa os números UID e GID.

O Linux gerencia os usuários e os grupos através de números conhecidos como UID (**U**ser **Id**entifier) e GID (**G**roup **Id**entifier). Como é possível perceber, UID são números de usuários e GID são números de grupos. Os nomes dos usuários e dos grupos servem apenas para facilitar o uso humano do computador.

Cada usuário precisa pertencer a um ou mais grupos. Como cada processo (e cada arquivo) pertence a um usuário, logo, esse processo pertence ao grupo de seu proprietário. Assim sendo, cada processo está associado a um UID e a um GID.

Os números UID e GID variam de 0 a 65536. Dependendo do sistema, o valor limite pode ser maior. No caso do usuário root, esses valores são sempre 0 (zero). Assim, para fazer com que um usuário tenha os mesmos privilégios que o root, é necessário que seu GID seja 0.

Sinais de processos

Os sinais são meios usados para que os processos possam se comunicar e para que o sistema possa interferir em seu funcionamento. Por exemplo, se o usuário executar o comando kill para interromper um processo, isso será feito por meio de um sinal.

Quando um processo recebe um determinado sinal e conta com instruções sobre o que fazer com ele, tal ação é colocada em prática. Se não houver instruções pré-programadas, o próprio Linux pode executar a ação de acordo com suas rotinas.

Entre os sinais existentes, tem-se os seguintes exemplos:

STOP - esse sinal tem a função de interromper a execução de um processo e só reativá-lo após o recebimento do sinal CONT;

CONT - esse sinal tem a função de instruir a execução de um processo após este ter sido interrompido;

SEGV - esse sinal informa erros de endereços de memória;

TERM - esse sinal tem a função de terminar completamente o processo, ou seja, este deixa de existir após a finalização;

ILL - esse sinal informa erros de instrução ilegal, por exemplo, quando ocorre divisão por zero;

KILL - esse sinal tem a função de "matar" um processo e é usado em momentos de criticidade.

O kill também é um comando que o usuário pode usar para enviar qualquer sinal, porém, se ele for usado de maneira isolada, ou seja, sem o parâmetro de um sinal, o kill por padrão executa o sinal TERM.

A sintaxe para a utilização do comando kill é a seguinte:

```
kill -SINAL PID
```

Como exemplo, vamos supor que você deseja interromper temporariamente a execução do processo de PID 4220. Para isso, pode-se usar o seguinte comando:

```
kill -STOP 4220
```

Para que o processo 4220 volte a ser executado, basta usar o comando:

```
kill -CONT 4220
```

Se o sinal precisa ser enviado a todos os processos, pode-se usar o número -1 no lugar do PID. Por exemplo:

```
kill -STOP -1
```

Como já dito, usar o comando kill isoladamente - por exemplo, kill 4220 - faz com que este use o sinal TERM por padrão. Esse sinal, no entanto, pode ser ignorado pelos processos. É por isso que é boa prática usar o comando "kill -9 PID" para "matar" um processo, pois o número nove representa o sinal kill e este não pode ser ignorado. Isso deixa claro que se você conhecer o número que é atribuído a um sinal, você pode usá-lo no lugar de seu nome. Com exceção de alguns sinais, a numeração de cada um pode mudar de acordo com a distribuição ou com a versão do kernel.

Também é comum usar o kill da seguinte forma: **kill -l PID**. A opção "-l" (letra L minúscula) é usada para listar os processos que aceitaram o kill.

Agora, imagine que você não saiba qual o PID de um processo e tenha se esquecido que o comando ps (visto mais à frente) descobre tal informação. Neste caso, pode-se usar o comando killall, desde que você saiba o nome do processo. A sintaxe é:

killall -SINAL processo

Por exemplo:

killall -STOP vi

Estado dos processos

Quando um processo é criado, isso não significa que ele será imediatamente executado. Além disso, determinados processos podem ser temporariamente paralisados para que o processador possa executar um processo prioritário. Isso quer dizer que os processos, em certos momentos, podem estar em situações de execução diferentes. O Linux trabalha, essencialmente, com quatro tipos de situação, isto é, estados:

Executável: o processo pode ser executado imediatamente;

Dormente: o processo precisa aguardar alguma coisa para ser executado. Só depois dessa "coisa" acontecer é que ele passa para o estado executável;

Zumbi: o processo é considerado "morto", mas, por alguma razão, ainda existe;

Parado: o processo está "congelado", ou seja, não pode ser executado.

Comandos nice e renice

Ao abordarmos os comandos nice e renice é necessário entender o conceito de gentileza. Um processo pode ter prioridade em relação a outros em sua execução. Quando um processo é gentil, significa que ele "oferece a gentileza" de permitir que um processo com prioridade maior que a sua seja executado antes dele. Os níveis de gentileza, também chamados de nice, são determinados através de números. Quanto mais alto for o valor nice, mais gentil é o processo. Geralmente, o intervalo de números usados no nice são os inteiros entre -19 e 19.

Embora determinar a prioridade de um processo não seja uma prática comum, afinal, o próprio Linux faz muito bem essa tarefa, isso pode ser necessário em alguma situação. Para isso, utiliza-se um comando que recebe o mesmo nome do conceito: nice. A sintaxe é:

nice -n prioridade processo

Por exemplo:

nice -n -5 ntpd

No exemplo, o ntpd recebe prioridade -5. Trata-se de uma prioridade alta, afinal, como já dito, quanto menor o número menor sua gentileza.

Se um determinado processo está em execução, isso acontece com uma prioridade já definida. Para alterar um processo nessa condição, usa-se o comando renice, cuja sintaxe é:

renice prioridade opção processo/destino

As opções do renice são:

-u - a alteração ocorrerá nos processos do usuário informado;

-g - a alteração ocorrerá nos processos do grupo indicado;

-p - a alteração ocorrerá no processo cujo PID for informado.

Um exemplo:

renice +19 1000 -u infowester

Neste caso, o comando renice alterou a prioridade do processo 1000, assim como a prioridade dos processos do usuário infowester.

Verificando processos com o ps

O ps é um comando de extrema importância para o gerenciamento de processos. Por ele, é possível saber quais os processos em execução atualmente, quais os UIDs e PIDs correspondentes, entre outros.

Se somente ps for digitado na linha de comando, geralmente o sistema mostra quais os processos do usuário. É preciso usar uma combinação de opções para obter mais detalhes.

As opções mais importantes são os seguintes:

a - mostra todos os processos existentes;

e - exibe as variáveis de ambiente relacionadas aos processos;

f - exibe a árvore de execução dos processos;

l - exibe mais campos no resultado;

m - mostra a quantidade de memória ocupada por cada processo;

u - exibe o nome do usuário que iniciou determinado processo e a hora em que isso ocorreu;

x - exibe os processos que não estão associados a terminais;

w - se o resultado de processo não couber em uma linha, essa opção faz com que o restante seja exibido na linha seguinte.

Das opções acima, a combinação mais usada (pelo menos aqui no Infowester) é aux:

```
$ ps
        PID TTY          TIME CMD
1508939 pts/0 00:00:00 bash
1510648 pts/0 00:00:00 ps
$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.1 167020  9980 ?        Ss   07:42   0:01 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    07:42   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   07:42   0:00 [rcu_gp]
…
hk       64378  0.0  1.4 2521352 114184 ?      Sl   07:43   0:01 /snap/firefox
```

```
hk        64407  0.4  2.4 2711400 198388 ?    Sl   07:43   0:06 /snap/firefox

hk        71935  0.0  0.7 387940 56936 ?      Sl   07:43   0:00 /snap/firefox
```

Note que usando a combinação lax, o resultado mostra mais detalhes:

```
$ ps lax

F   UID          PID    PPID PRI  NI   VSZ   RSS WCHAN  STAT TTY        TIME COMMAND

4         0        1        0  20    0 167020  9980 -        Ss   ?          0:01 /sbin/init splash

1         0        2        0  20    0        0        0 -        S        ?          0:00 [kthreadd]

1         0        3        2   0 -20        0        0 -        I<   ?          0:00 [rcu_gp]

…

4  1000   13756   13161  20    0 34325240 352452 do_pol Sl ?        0:24
/snap/brave/202/opt/brave.com/brave/brave

0  1000   14326          5032  20    0 33588764 3528 ep_pol Sl  ?              0:00
/snap/brave/202/opt/brave.com/brave/chrome_crashpad_handler --

0  1000   14338          5032  20    0 33580552 3040 ep_pol Sl  ?              0:00
/snap/brave/202/opt/brave.com/brave/chrome_crashpad_handler --
```

A seguir, segue a descrição dos campos mostrados anteriormente e alguns que só são mostrados com a combinação lax:

USER - nome do usuário dono do processo;

UID - número de identificação do usuário dono do processo;

PID - número de identificação do processo;

PPID - número de identificação do processo pai;

%CPU - porcentagem do processamento usado;

%MEM - porcentagem da memória usada;

VSZ - indica o tamanho virtual do processo;

RSS - sigla de **R**esident **S**et **S**ize, indica a quantidade de memória usada (em KB);

TTY - indica o identificador do terminal do processo;

START - hora em que o processo foi iniciado;

TIME - tempo de processamento já consumido pelo processo;

COMMAND - nome do comando que executa aquele processo;

PRI - valor da prioridade do processo;

NI - valor preciso da prioridade (geralmente igual aos valores de PRI);

WCHAN - mostra a função do kernel onde o processo se encontra em modo suspenso;

STAT - indica o estado atual do processo, sendo representado por uma letra: **R** - executável; **D** - em espera no disco; **S** - Suspenso; **T** - interrompido; **Z** - Zumbi. Essas letras podem ser combinadas e ainda acrescidas de: **W** - processo paginado em disco; **<** - processo com prioridade maior que o convencional; **N** - processo com prioridade menor que o convencional; **L** - processo com alguns recursos bloqueados no kernel.

Verificando processos com o top

O comando ps trabalha como se tirasse uma fotografia da situação dos processos naquele momento. O comando top, por sua vez, coleta as informações, mas as atualiza regularmente. Geralmente essa atualização ocorre a cada 10 segundos.

A sintaxe do comando top é a seguinte:

top -opção

Entre as opções, tem-se as que se seguem:

-d - atualiza o top após um determinado período de tempo (em segundos). Para isso, informe a quantidade de segundos após a letra d. Por exemplo: top -d 30;

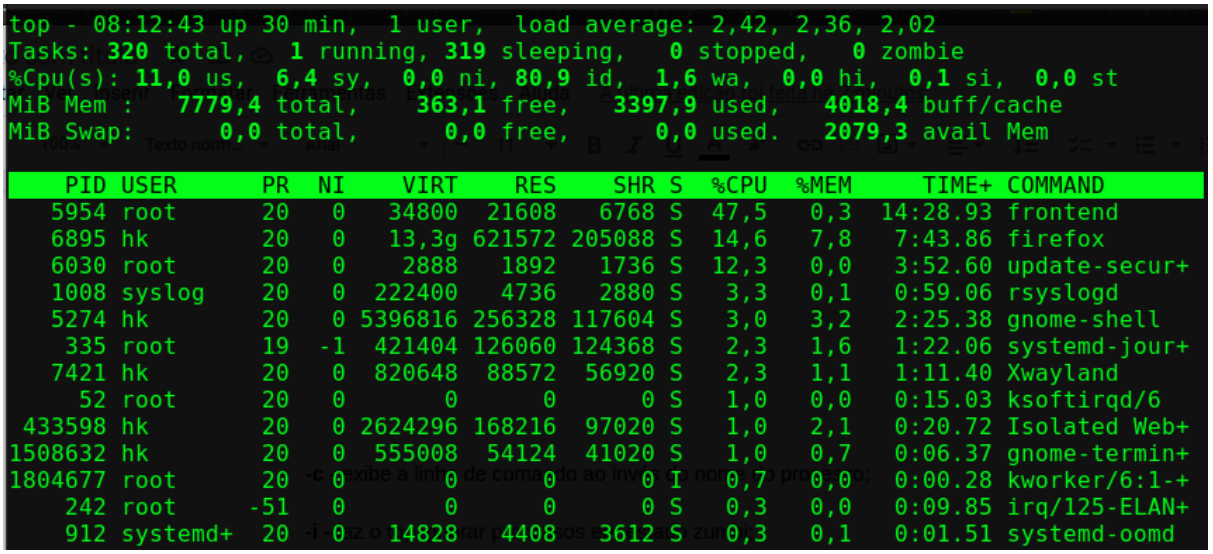
-c - exibe a linha de comando ao invés do nome do processo;

-i - faz o top ignorar processos em estado zumbi;

-s - executa o top em modo seguro.

É possível manipular alguns recursos do comando top através das teclas do teclado. Por exemplo, para atualizar imediatamente o resultado exibido, basta pressionar a tecla de espaço. Se pressionar a tecla q, o top é finalizado. Pressione a tecla h enquanto estiver utilizando o top para ver a lista completa de opções e teclas de atalho.

$ top



Para sair do “top” basta usar a tecla “q” de quit (encerrar).

Os recursos jobs, fg e bg, fuser, pstree, nohup

Para ter ainda mais controle sobre os processos executados no Linux, pode-se utilizar os seguintes comandos: jobs, fg e bg, fuser, pstree, nohup. Cada um é descrito a seguir:

jobs - serve para visualizar os processos que estão parados ou executando em segundo plano (background). Quando um processo está nessa condição, significa sua execução é feita pelo kernel sem que esteja vinculada a um terminal. Em outras palavras, um processo em segundo plano é aquele que é executado enquanto o usuário faz outra coisa no sistema. Uma dica para saber se o processo está em background é verificar a existência do

caractere & no final da linha. Se o processo estiver parado, geralmente a palavra "stopped" aparece na linha, do contrário, a palavra "running" é exibida. A sintaxe do jobs é:

$ jobs [opção]

As opções disponíveis são:

-l - lista os processos através do PID;

-r - lista apenas os processos em execução;

-s - lista apenas os processos parados.

Se na linha de um processo aparecer o sinal positivo (+), significa que este é o processo mais recente a ser paralisado ou a estar em segundo plano. Se o sinal for negativo (-), o processo foi o penúltimo. Note também que no início da linha um número é mostrado entre colchetes. Muitos confundem esse valor com o PID do processo, mas, na verdade, trata-se do número de ordem usado pelo jobs.

fg e bg: o fg é um comando que permite a um processo em segundo plano (ou parado) passar para o primeiro (foreground), enquanto que o bg passa um processo do primeiro plano para o segundo. Para usar o bg, deve-se paralisar o processo. Isso pode ser feito pressionando-se as teclas Ctrl + Z no teclado. Em seguida, digita-se o comando da seguinte forma:

bg +número

O número mencionado corresponde ao valor de ordem informado no início da linha quando o comando jobs é usado.

Quanto ao comando fg, a sintaxe é a mesma:

fg +número

fuser: o comando fuser mostra qual processo faz uso de um determinado arquivo ou diretório. Sua sintaxe é:

fuser -opção caminho (do arquivo ou diretório)

Entre as opções, tem-se:

-k - finaliza o processo que utiliza o arquivo/diretório em questão;

-i - deve ser usada em conjunto com a opção k e serve para perguntar se a finalização do processo deve ser feita;

-u - mostra o proprietário do processo;

-v - o resultado é mostrado em um padrão de exibição semelhante ao comando ps.

pstree: esse comando mostra processos relacionados em formato de árvore. Sua sintaxe é:

pstree -opção PID

Entre as opções, tem-se:

-u - mostra o proprietário do processo;

-p - exibe o PID após o nome do processo;

-c - mostra a relação de processos ativos;

-G - usa determinados caracteres para exibir o resultado em um formato gráfico.

Um detalhe importante: se ao digitar o comando pstree o PID não for informado, todos os processos serão listados.

nohup: o comando nohup possibilita ao processo ficar ativo mesmo quando o usuário faz logout. É da natureza dos sistemas baseados em Unix interromper processos caso seu proprietário não esteja mais ativo, por isso, o nohup pode ser muito útil. Sua sintaxe é:

$ nohup [comando]