Comandos para gerência de processos no Linux

Processo é uma instância de um software ou um código ou pedaço dele em execução, ou seja, um programa que está sendo processado (executado) pelo processador. Os processos geralmente são chamados de tarefas (tasks).
O Linux se trata de um sistema operacional multitarefa e desta forma possui recursos (comandos) para gerenciar os processos de forma bastante interativa, algo transparente na maioria dos outros sistemas operacionais a exemplo do Windows e MacOS.

Escalonamento de Processos no Linux
É importante entender como o kernel do Linux realiza o escalonamento de processos:
- Cada processador (core ou núcleo) só executa um processo (tarefa) por vez.
- Time sharing: como cada processador só executa uma tarefa por vez, o que o kernel faz é escalonar (ou rotacionar) os processos entre as CPUs (ou cores) disponiveis, definindo que cada processo utilize a CPU por um determinado tempo (quantum);
- O escalonamento (rotação) é feito automaticamente pelo kernel, que se baseia nos tipos de processos em execução.

Tipos de Processos
De maneira geral, podemos classificar os processos nos seguintes tipos:

- Convencional: Processos normais do sistema.
  - O kernel dá a esses processos os valores de prioridades de 100 a 139, sendo 120 o padrão. Sendo 100 significa maior prioridade, ou seja, terá mais tempo de uso da CPU, maior time sharing. Nesse caso 139 é a menor prioridade, menor tempo.

- Tempo Real: Processos que precisam ser executados imediatamente, não obedecendo o escalonamento de processos.
  - Normalmente são processos internos do kernel.
  - Esses processos recebem as prioridades de 1 a 99, e é comum que seja representado pelo código “rt”, de real-time.

Prioridade (Priority ou PRI) e Nice (NI)
Diferença entre Prioridade (PRI) e Nice (NI), que podem ser verificados via comendo "top":
- A prioridade (PRI) de um processo é definida automaticamente e dinâmicamente pelo kernel Linux, assumindo os valores que citei acima;
- O Nice (NI) é um atributo que permite ao administrador ou usuário influenciar a prioridade do processo. Quando usamos os comandos nice e renice para definir esse atributo, estamos definindo um NICE que irá consequentemente impactar a prioridade. Por padrão, o NICE de um processo é 0.

OBS:
O comando "top" reduz o número PRI e NI, assim, exibe 20 ao invés de 120, 9 ao invés de 109, e assim por diante.

OBS:
Pode-se verificar a prioridade de um processos através do "sched", buscando o termo “prio”:
cat /proc/<PID>/sched

Exemplo:
$ nice -n10 firefox &
[1] 132116

$ grep prio /proc/132116/sched
prio : 131

$ top
```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 132116 hk        31  11 3497168 351428 119004 S  29,4   4,4   2:19.95 firefox
```

Assim, temos que percebe-se que:
PR = 31 = 131
NI = 11

Comandos de processos no Linux
Alguns dos principais comandos de gerenciamento de processos no Linux estão descritos a seguir.

1. top
O comando top é a maneira mais comum de verificar o uso de processos do sistema e constatar quais deles estão consumindo mais memória ou processamento. Note que os primeiros itens da lista são os que mais consomem recursos do computador. Para cancelar a execução e voltar à linha de comando, basta pressionar a tecla Q ou a combinação Ctrl+C.
Importante nessa relação é a primeira coluna, PID, que exibe o número de identificação de determinado processo. É por meio desse número que você poderá, por exemplo, encerrar um processo com o comando kill.
Exemplo:
```
top - 13:29:14 up 6 days,  3:34,  1 user,  load average: 0,67, 0,64, 0,70
Tasks: 348 total,   1 running, 347 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,3 us,  1,5 sy,  0,0 ni, 95,1 id,  0,0 wa,  0,0 hi,  0,1 si,  0,0 st
MiB Mem :   7779,4 total,    136,0 free,   5448,3 used,   2195,1 buff/cache
MiB Swap:      0,0 total,      0,0 free,      0,0 used.   1081,7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  80444 hk        20   0 1125,4g 308256 112016 S  12,6   3,9  51:44.52 brave
   5633 hk        20   0   32,9g 244840 176412 S   4,0   3,1 387:32.62 brave
   2108 hk        20   0 6599584 504908  48396 S   2,7   6,3 133:02.66 gnome-shell
 127196 hk        20   0  553504  52464  40056 S   1,3   0,7   0:05.33 gnome-termi+
 127267 hk        20   0   13336   4324   3348 R   1,0   0,1   0:00.83 top
 119835 hk        20   0 3359952 595420  83796 S   0,7   7,5   3:55.82 Isolated We+
     14 root      20   0       0      0      0 I   0,3   0,0   3:17.63 rcu_sched
```

2. htop
Usado para fornecimento de detalhes acerca dos processos em execução. A diferença entre eles, no entanto, é a maneira como a qual são apresentadas as informações: ela ocorre de forma mais intuitiva e inteligente.
No htop, cores são utilizadas para facilitar a compreensão do usuário. Além disso, ao digitar F5 a apresentação adquire o formato de diagrama de árvore.
O htop é integrado ao comando kill, e permite fechar processos usando a tecla F9.

Obs: o htop não vem pré-instalado em todas as ditribuições, portanto, caso não esteja instalado, é necessário usar o comando:
$ sudo apt install htop
[sudo] password for aluno:

3. ps

O comando ps lista os processos em execução no sistema. Porém, diferentemente do top, ele não traz informações sobre o quanto de processamento ou de memória ele está consumindo. Apesar disso, o ps é uma maneira bem mais ágil de consultar o PID de um processo, principalmente ao ser usado em conjunto com o grep.

Para saber qual é o PID do vim, por exemplo, um usuário poderia executar ps aux | grep -i vim. Antes de de executá-lo, no entanto, vamos entender o que faz cada parte desse comando: as opções aux garantem que o ps exiba processos de todos os usuários (a), o nome do usuário responsável pelo processo (u) e também aqueles processos que não estão, necessariamente, sendo executados naquele terminal (x). A barra vertical, ou pipe (|), faz com que o resultado seja direcionado para o comando grep que, por sua vez filtrará apenas as linhas que tenham a palavra vim.

Fique à vontade para modificar essa linha e substituir vim pelo nome da aplicação que você estiver procurando.

Exemplo:

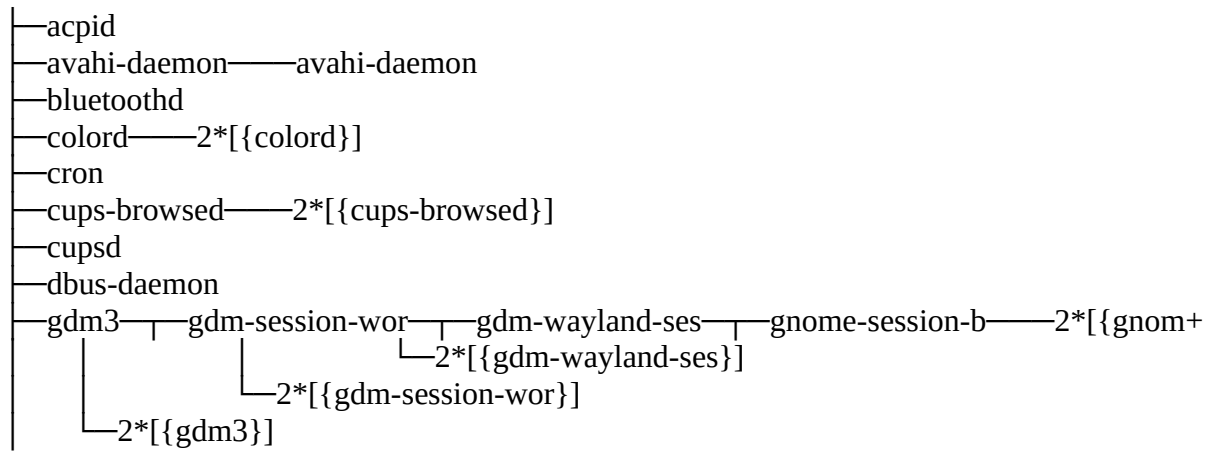
```
$ ps
    PID TTY          TIME CMD
 127214 pts/0    00:00:00 bash
 127227 pts/0    00:00:00 ps

$ ps -e -o uid,pid,ppid,pri,ni,cmd
  UID     PID    PPID PRI  NI CMD
    0       1       0  19   0 /sbin/init splash
    0       2       0  19   0 [kthreadd]
    0       3       2  39 -20 [rcu_gp]
    0       4       2  39 -20 [rcu_par_gp]
    0       5       2  39 -20 [netns]
    0       7       2  39 -20 [kworker/0:0H-kblockd]
    ...

$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 168368  9776 ?        Ss   set08   0:12 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    set08   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   set08   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   set08   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   set08   0:00 [netns]
root           7  0.0  0.0      0     0 ?        I<   set08   0:00 [kworker/0:0H
root          10  0.0  0.0      0     0 ?        I<   set08   0:00 [mm_percpu_wq
root          11  0.0  0.0      0     0 ?        S    set08   0:00 [rcu_tasks_ru
root          12  0.0  0.0      0     0 ?        S    set08   0:00 [rcu_tasks_tr
...
```

4. pstree

Também há uma forma de visualizar os processos em forma de árvore, tornando mais visível as relações entre eles. Para isso, basta usar o comando pstree.

Exemplo:

```
$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─accounts-daemon───2*[{accounts-daemon}]
```



...

5. kill
Se um software travou ou precisa ser interrompido de qualquer forma, o kill é a solução. Basta executar)o comando seguido do PID do processo para que a aplicação "morra". Se mesmo depois disso você perceber que o processo ainda existe, tente acrescentar a opção -9 ao comando: kill -9 PID. Assim força o processo a ser interrompido a qualquer custo.
Se quiser matar mais de um processo ao mesmo tempo, basta listar os PIDs separando-os com um espaço, logo depois do comando kill.
Exemplo:
$ kill -l (lista todos os sinais do comando kill)
$ kill 3657 6785 3456 (mata os processos listados pelos seus PID)
$ kill -11 1052 (encerra o processo pelo seu PID)
$ kill -9 1052 (caso o -11 não consiga encrrar o processo, o -9 forçar a parada obrigatória de um processo)

6. killall
Caso prefira, você também pode matar de uma vez só todos os comandos selecionado ao nome da aplicação, assim, basta usar o comando killall seguido do nome da aplicação..
Porém, o killall exige uma certa rigidez ao informar o nome do processo. Caso você não tenha certeza do nome completo, pode tentar o pkill, que faz diversas associações com a palavra-chave digitada.
Exemplo:
$ killall -l (lista todos os sinais do comando killall)

7. pkill
O comando pkill é análogo ao comando killall. Ele pode enviar sinais para os processos e recebe como parâmetro uma expressão ou nome do processo.
Ele é usado geralmente para terminar a execução de processos que possuem diversos processos filhos executando em simultâneo.

Exemplo:
Abri duas janelas de navegador Firefox, o Chrome, por exemplo, cada um cum duas abas.
$ pkill firefox (executar o pkill em um terminal)
ou
$ pkill chrome

8. nice
O comando nice ajusta o tempo disponível de CPU de um processo para mais ou para menos prioridade.
Se o ajuste "nice" para um processo for um número positivo, significa que o processo está ajustado para um prioridade de execução mais baixa, ou seja, está sendo solicitado ao sistema operacional que o execute com uma prioridade de execução menor.
Se o ajuste "nice" de um processo for um número negativo, significa que o programa está solicitando prioridade maior para ser executado, para passar à frente de outros processos na CPU.
O ajuste de prioridade "nice" vai do –20 (maior prioridade) até o 19 (menor prioridade).
Se não for passado nenhum valor de ajuste no "nice", o comando ajustará a prioridade para o padrão +10.
Exemplo:
$ nice chrome (neste caso o chrome terá prioridade de execução padrão +10)
$ nice –n –10 firefox (neste caso o firefox terá prioridade de execução -10)

9. renice
Todos os processos do Linux possuem prioridades de execução, variando em uma escala que vai de 19 (menos significativa) a -20 (mais significativa). Por padrão, os processos executados por um usuário ganham a prioridade 0, mas por meio do comando renice é possível alterar esse valor para algum nível entre 0 e 19. Apenas o usuário administrador (root) é capaz de ir além, alterando prioridades de qualquer processo e chegando até o nível máximo de -20.
Para realizar esse tipo de operação, basta seguir a sintaxe renice novaprioridade -p PID. Se quiséssemos dar mais prioridade a um processo de PID 1516, por exemplo, usaríamos: sudo renice -10 -p 1516. Lembre-se que o sudo exigirá a senha do seu usuário antes de executar o comando.
Exemplo:

OBS:
Diferença do "nice" e "renice":
- nice: utilizado para iniciar processos, que ainda não foram iniciados ainda, com prioridade diferente de sua prioridade padrão, portanto, usado para alterar a prioridade antes de iniciar um processo.
- renice: utilizado em processos que já estão na CPU executando e ou esperando para executar, assim, é usado para alterar prioridade de processos que já foram iniciados.

10. ionice
O comando ionice atribui maior ou menor prioridade na leitura e escrita no disco. Ou seja, diferentemente de “nice” e “renice”, que trabalham com prioridade sobre CPU e Memória, o comando “ionice” trabalha com prioridade de leitura e escrita.
Exemplo:
$ ionice -c 3 -p 2536
$ ionice -c 2 -p 2536
$ ionice -c 1 -p 2536

10. Ctrl + c
A combinação das teclas Ctrl + c finaliza um processo.

11. Ctrl + z
A combinação das teclas Ctrl + c suspende um processo, deixando-o em standby em segundo plano (background).

12. foreground - fg
Trás um processo para foreground (fg), primeiro plano, e o executa no processador.
Exemplo:

```
$ top
top - 13:39:45 up 6 days,  3:44,  1 user,  load average: 0,76, 0,60, 0,64
Tasks: 350 total,   1 running, 347 sleeping,   2 stopped,   0 zombie
%Cpu(s):  3,9 us,  2,6 sy,  0,0 ni, 93,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7779,4 total,    174,1 free,   5455,2 used,   2150,2 buff/cache
MiB Swap:      0,0 total,      0,0 free,      0,0 used.   1077,0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 127762 hk        20   0   13372   4168   3292 R  16,7   0,1   0:00.05 top
   2108 hk        20   0 6596544 510640  53316 S  11,1   6,4 133:27.38 gnome-shell
  80444 hk        20   0 1125,4g 307004 111608 S  11,1   3,9  52:46.97 brave
   5633 hk        20   0   32,9g 243280 174892 S   5,6   3,1 388:43.16 brave
      1 root      20   0  168368   9720   4220 S   0,0   0,1   0:12.72 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.17 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
      7 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0+
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_r+
     12 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_t+
     13 root      20   0       0      0      0 S   0,0   0,0   0:03.34 ksoftirqd/0
     14 root      20   0       0      0      0 I   0,0   0,0   3:18.22 rcu_sched
     15 root      rt   0       0      0      0 S   0,0   0,0   0:00.91 migration/0
     16 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject+
[3]+  Stopped                 top

$ fg 3
top - 13:39:45 up 6 days,  3:44,  1 user,  load average: 0,76, 0,60, 0,64
Tasks: 350 total,   1 running, 347 sleeping,   2 stopped,   0 zombie
%Cpu(s):  3,9 us,  2,6 sy,  0,0 ni, 93,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7779,4 total,    174,1 free,   5455,2 used,   2150,2 buff/cache
MiB Swap:      0,0 total,      0,0 free,      0,0 used.   1077,0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 127762 hk        20   0   13372   4168   3292 R  16,7   0,1   0:00.05 top
   2108 hk        20   0 6596544 510640  53316 S  11,1   6,4 133:27.38 gnome-shell
  80444 hk        20   0 1125,4g 307004 111608 S  11,1   3,9  52:46.97 brave
   5633 hk        20   0   32,9g 243280 174892 S   5,6   3,1 388:43.16 brave
      1 root      20   0  168368   9720   4220 S   0,0   0,1   0:12.72 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.17 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
```

```
      7 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0+
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_r+
     12 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_tasks_t+
     13 root      20   0       0      0      0 S   0,0   0,0   0:03.34 ksoftirqd/0
     14 root      20   0       0      0      0 I   0,0   0,0   3:18.22 rcu_sched
     15 root      rt   0       0      0      0 S   0,0   0,0   0:00.91 migration/0
     16 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject+
...
```

13. background - bg

Faz um processo ir para background (bg), segundo plano, e tira o processo da fila de processos e o leva para fila de suspensos, até que seja novamente executado fg.

Exemplo:

```
$ bg 3
[3]+ top &

[3]+  Stopped                 top
```

14. ionice

O comando ionice atribui maior ou menor prioridade na leitura e escrita no disco, ou seja, diferentemente de “nice” e “renice”, que trabalham com prioridade sobre CPU e Memória, o comando “ionice” trabalha com prioridade de leitura e escrita.

As prioridades do comando ionice são:

-c 3 -> Iddle: só executa se houver recurso livre;

-c 2 -> Best-Enffort: executa sem consumir muito recurso;

-c 1 -> Real Time: executa com prioridade.

```
$ ionice -c 3 -p 132116
$ ionice -c 2 -p 132116
$ ionice -c 1 -p 132116
```