

Experiência 05 – Escalonamento de processos em Linux

Introdução

O problema básico de escalonamento em sistemas operacionais é como satisfazer simultaneamente objetivos conflitantes: tempo de resposta rápido, bom *throughput* para processos background, evitar postergação indefinida, conciliar processos de alta prioridade com de baixa prioridade, etc. O conjunto de regras utilizado para determinar como, quando e qual processo deverá ser executado é conhecido como política de escalonamento. Nesta seção, nós estudaremos como o Linux implementa seu escalonador e qual a política empregada para determinar quais processos recebem o processador.

Tradicionalmente, os processos são divididos em três grandes classes: processos interativos, processos batch e processos tempo real. Em cada classe, os processos podem ser ainda subdivididos em *I/O bound* ou *CPU bound* de acordo com a proporção de tempo que ficam esperando por operações de entrada e saída ou utilizando o processador. O escalonador do Linux não distingue processos interativos de processos batch, diferenciando-os apenas dos processos tempo real. Como todos os outros escalonadores UNIX, o escalonador Linux privilegia os processos *I/O bound* em relação aos *CPU bound* de forma a oferecer um melhor tempo de resposta às aplicações interativas.

O escalonador do Linux é baseado em time-sharing, ou seja, o tempo do processador é dividido em fatias de tempo (quantum) as quais são alocadas aos processos. Se, durante a execução de um processo, o quantum é esgotado, um novo processo é selecionado para execução, provocando então uma troca de contexto. Esse procedimento é completamente transparente ao processo e baseia-se em interrupções de tempo. Esse comportamento confere ao Linux um escalonamento do tipo preemptivo.

O algoritmo de escalonamento do Linux divide o tempo de processamento em épocas (*epochs*). Cada processo, no momento de sua criação, recebe um quantum calculado no início de uma época. Diferentes processos podem possuir diferentes valores de quantum. O valor do quantum corresponde à duração da época, e essa, por sua vez, é um múltiplo de 10 ms inferior a 100 ms.

Outra característica do escalonador Linux é a existência de prioridades dinâmicas. O escalonador do Linux monitora o comportamento de um processo e ajusta dinamicamente sua prioridade, visando a equalizar o uso do processador entre os processos. Processos que recentemente ocuparam o processador durante um período de tempo considerado “longo” têm sua prioridade reduzida. De forma análoga, aqueles que estão há muito tempo sem executar recebem um aumento na sua prioridade, sendo então beneficiados em novas operações de escalonamento.

Na realidade, o sistema Linux trabalha com dois tipos de prioridades: estática e dinâmica. As prioridades estáticas são utilizadas exclusivamente por processos de tempo real correspondendo a valores na faixa de 1-99. Nesse caso, a prioridade do processo tempo real é definida pelo usuário e não é modificada pelo escalonador. Somente usuários com privilégios especiais têm a capacidade de criar e definir processos tempo real. O esquema de prioridades dinâmicas é aplicado aos processos interativos e batch. Aqui, a prioridade é calculada, considerando-se a prioridade base do processo e a quantidade de tempo restante em seu quantum.

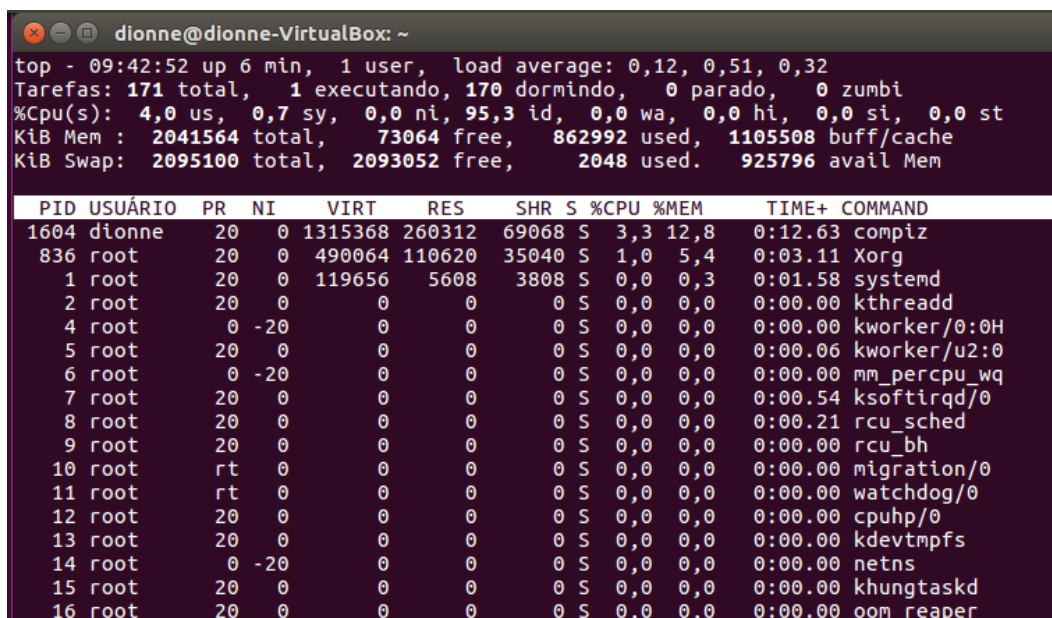
O escalonador do Linux executa os processos de prioridade dinâmica apenas quando não há processos de tempo real. Em outros termos, os processos de prioridade estática recebem uma prioridade maior que os processos de prioridade dinâmica. Para selecionar um processo para execução, o escalonador do Linux prevê três políticas diferentes:

- **SCHED_FIFO**: Essa política é válida apenas para os processos de tempo real. Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade. Nessa política, quando um processo é alocado ao processador, ele executa até que uma de três situações ocorra: (i) um processo de tempo real de prioridade superior torna-se apto a executar; (ii) o processo libera espontaneamente o processador para processos de prioridade igual à sua; (iii) o processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização.
- **SCHED_RR**: Na criação, o descritor do processo é inserido no final da fila correspondente à sua prioridade. Quando um processo é alocado ao processador, ele executa até que uma de quatro situações ocorra: (i) seu período de execução (quantum) tenha se esgotado nesse caso o processo é inserido no final de sua fila de prioridade; (ii) um processo de prioridade superior torna-se apto a executar; (iii) o processo libera espontaneamente o processador para processos de prioridade igual a sua; (iv) o processo termina, ou bloqueia-se, em uma operação de entrada e saída ou de sincronização. Essa política também só é válida para processos de tempo real.
- **SCHED_OTHER**: Corresponde a um esquema de filas multinível de prioridades dinâmicas com *timesharing*. Os processos interativos e *batch* recaem nessa categoria.

Fonte: <http://www.inf.ufrgs.br/~asc/livro/secao94.pdf>

Atividades

- 1) Execute o comando **top** para verificar a utilização da memória e da CPU pelos processos executados no SO Linux.



```

top - 09:42:52 up 6 min,  1 user,  load average: 0,12, 0,51, 0,32
Tarefas: 171 total,  1 executando, 170 dormindo,  0 parado,  0 zumbi
%Cpu(s):  4,0 us,  0,7 sy,  0,0 ni, 95,3 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 2041564 total,  73064 free,  862992 used, 1105508 buff/cache
KiB Swap: 2095100 total, 2093052 free,  2048 used.  925796 avail Mem

  PID  USUÁRIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  TIME+  COMMAND
1604 dionne   20   0 1315368 260312 69068 S   3,3 12,8   0:12.63 compiz
836 root      20   0 490064 110620 35040 S   1,0  5,4   0:03.11 Xorg
1 root   20   0 119656 5608 3808 S   0,0  0,3   0:01.58 systemd
2 root   20   0      0      0      0 S   0,0  0,0   0:00.00 kthreadd
4 root    0 -20      0      0      0 S   0,0  0,0   0:00.00 kworker/0:0H
5 root   20   0      0      0      0 S   0,0  0,0   0:00.06 kworker/u2:0
6 root    0 -20      0      0      0 S   0,0  0,0   0:00.00 mm_percpu_wq
7 root   20   0      0      0      0 S   0,0  0,0   0:00.54 ksoftirqd/0
8 root   20   0      0      0      0 S   0,0  0,0   0:00.21 rcu_sched
9 root   20   0      0      0      0 S   0,0  0,0   0:00.00 rcu_bh
10 root  rt    0      0      0      0 S   0,0  0,0   0:00.00 migration/0
11 root  rt    0      0      0      0 S   0,0  0,0   0:00.00 watchdog/0
12 root   20   0      0      0      0 S   0,0  0,0   0:00.00 cpuhp/0
13 root   20   0      0      0      0 S   0,0  0,0   0:00.00 kdevtmpfs
14 root    0 -20      0      0      0 S   0,0  0,0   0:00.00 netns
15 root   20   0      0      0      0 S   0,0  0,0   0:00.00 khungtaskd
16 root   20   0      0      0      0 S   0,0  0,0   0:00.00 oom_reaper

```

- 2) A grande maioria dos processos no SO são criados com a mesma prioridade, conforme observado na coluna PR na execução do comando **top** acima. Para alterar a prioridade de um processo no SO Linux alteramos o valor NI (**nice**) que varia entre -20 e +19. Processos com NI -20 possuem prioridade máxima, enquanto processos com NI 19 possuem prioridade mínima na alocação de tempo de CPU (slice time).
- 3) O comando **nice** cria um novo processo alterando a sua prioridade, enquanto o comando **renice** altera a prioridade de um processo em tempo de execução.
 - a. Alterando prioridade de processos a serem criados:

- i. **nice -15 nome_do_processo**
 - ii. **nice -15 apt-get install eclipse**
- b. Alterando a prioridade de processos em execução
 - i. **renice 10 -p número_do_processo**
 - ii. **renice 10 -p 1604**

Tarefa 01

- 1) Abra duas janelas de terminal;
- 2) Em uma janela do terminal execute o comando **top**;
- 3) Na outra janela execute o comando **renice** escolhendo um PID de algum processo para ser alterado em a sua prioridade em +10 e posteriormente em – 10;
- 4) Execute o comando **nice -20 apt-get upgrade**;
- 5) Para os passos 3 e 4 capture as telas do terminal que executa o comando **top**;
- 6) Descreva o comportamento observado com o comportamento previsto.

Tarefa 02

Implementar 02 (dois) escalonadores de processos: Round Robin e Fila única com prioridades (Linux). Alguns requisitos que precisam ser atendidos pelo simulador:

- 1) O escalonador de processo que utiliza o Round Robin deve ter as seguintes entradas:
 - a. Time slice da CPU;
 - b. Processos com seus respectivos tempos de CPU e, caso necessário, com a sua prioridade, que deverá estar entre 0 e 10, sendo 0 processo com maior prioridade.

Time slice: 10

P1, 200

P2, 300

P3, 25

- 2) A saída do escalonador deverá ser o processo que está usando a CPU e o tempo ocupado na CPU;

CPU: P1, 10; P2, 10; P3, 10;

P1, 10; P2, 10; P3, 10;

P1, 10; P2, 10; P3, 5;*

P1, 10; P2, 10;

P1, 10; P2, 10;

- 3) No Round Robin um processo saído da CPU que não foi finalizado deverá entrar no final de fila e aguardar a próxima vez que deverá ocupar a CPU. Quando o processo for finalizado, deverá ser indicado através do caractere *.
- 4) No algoritmo com prioridade, os processos devem ser reconduzidos para a sua faixa de prioridade, sendo colocado no final da sua faixa de prioridade quando o tempo de CPU não for suficiente para sua execução. Quando o processo for finalizado, deverá ser indicado através do caractere *.