103.6 Modificar a prioridade de execução de um processo

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.6

Peso

2

Áreas chave de conhecimento

- Saber a prioridade padrão de um processo que é criado.
- Executar um programa com maior ou menor prioridade do que o padrão.
- Mudar a prioridade de um processo em execução.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `nice`
- `ps`
- `renice`
- `top`

103.6 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.6 Modificar prioridades de execução de processos
Lição:	1 de 1

Introdução

Os sistemas operacionais capazes de executar mais de um processo ao mesmo tempo são chamados de sistemas multitarefa ou multiprocessamento. Embora a verdadeira simultaneidade só aconteça quando há mais de uma unidade de processamento disponível, os sistemas de processador único podem imitar a simultaneidade alternando entre os processos muito rapidamente. Essa técnica também é empregada em sistemas com muitas CPUs equivalentes, ou sistemas de *multiprocessamento simétrico (SMP)*, dado que o número de processos concorrentes em potencial excede em muito o número de unidades de processamento disponíveis.

Na verdade, apenas um processo por vez pode controlar a CPU. No entanto, a maioria das atividades processadas são *chamadas do sistema*, ou seja, o processo em execução transfere o controle da CPU para um processo do sistema operacional para que ele execute a operação solicitada. As chamadas do sistema são responsáveis por toda a comunicação entre dispositivos, como alocações de memória, leitura e gravação em sistemas de arquivos, impressão de texto na tela, interação com o usuário, transferências de rede, etc. Transferir o controle da CPU durante uma chamada do sistema permite que o sistema operacional decida se deve devolver o controle da CPU para o processo anterior ou transferi-lo para outro processo. Como as CPUs modernas são capazes de executar instruções muito

mais rápido do que a maioria dos hardwares externos pode se comunicar entre si, um novo processo de controle pode fazer boa parte do trabalho da CPU enquanto as respostas de hardware solicitadas anteriormente ainda não estão disponíveis. Para garantir o aproveitamento máximo da CPU, os sistemas operacionais de multiprocessamento mantêm uma fila dinâmica de processos ativos aguardando um slot de tempo da CPU.

Embora elas permitam melhorar significativamente a utilização do tempo da CPU, confiar apenas nas chamadas do sistema para alternar entre os processos não basta para obter um desempenho multitarefa satisfatório. Um processo que não faz chamadas de sistema poderia controlar a CPU indefinidamente. Por essa razão, os sistemas operacionais modernos também são *preventivos*, ou seja, um processo em execução pode ser posto de volta na fila para que um processo mais importante assuma o controle da CPU, mesmo que o processo em execução não tenha feito uma chamada de sistema.

O Agendador do Linux

O Linux, sendo um sistema operacional de multiprocessamento preventivo, implementa um agendador que organiza a fila de processos. Mais precisamente, o agendador também decide qual *thread* da fila será executada — um processo pode ter muitas threads independentes — mas processo e thread são termos intercambiáveis neste contexto. Cada processo tem dois predicados que intervêm em seu agendamento: a *política de programação* e a *prioridade de programação*.

Existem dois tipos principais de políticas de programação: *políticas em tempo real* e *políticas normais*. Os processos em uma política em tempo real são programados diretamente por seus valores de prioridade. Se um processo mais importante estiver pronto para ser executado, um processo menos importante será interrompido e o processo de prioridade mais alta assumirá o controle da CPU. Um processo de prioridade mais baixa obterá o controle da CPU apenas se os processos de prioridade mais alta estiverem ociosos ou aguardando a resposta do hardware.

Qualquer processo em tempo real tem mais prioridade do que um processo normal. Como um sistema operacional de uso geral, o Linux executa apenas um punhado de processos em tempo real. A maioria dos processos, incluindo o sistema e os programas, são executados sob as políticas de programação normais. Os processos normais geralmente têm o mesmo valor de prioridade, mas as políticas normais podem definir regras de prioridade de execução usando outro predicado do processo: o *valor nice*. Para evitar confusão com as prioridades dinâmicas derivadas de valores nice, as prioridades de programação são geralmente chamadas de prioridades *estáticas*.

O agendador do Linux pode ser configurado de muitas maneiras diferentes e existem formas ainda mais intrincadas de estabelecer prioridades, mas esses conceitos gerais sempre se aplicam. Ao inspecionar e refinar a programação do processo, é importante ter em mente que apenas os processos sob a política de programação normal serão afetados.

Como ler as prioridades

O Linux reserva as prioridades estáticas de 0 a 99 para processos em tempo real. As prioridades estáticas de 100 a 139 são atribuídas a processos normais, o que significa que existem 39 níveis de prioridade diferentes para os processos normais. Valores mais baixos indicam uma prioridade mais alta. A prioridade estática de um processo ativo pode ser encontrada no arquivo `sched`, localizado em seu diretório respectivo dentro do sistema de arquivos `/proc`:

```
$ grep ^prio /proc/1/sched
prio                  :       120
```

Como mostrado no exemplo, a linha que começa com `prio` fornece o valor de prioridade do processo (o processo PID 1 é o processo *init* ou *systemd*, o primeiro processo que o kernel inicia durante a inicialização do sistema). A prioridade padrão para os processos normais é 120, podendo assim ser diminuída para 100 ou aumentada para 139. As prioridades de todos os processos em execução podem ser verificadas com o comando `ps -Al` ou `ps -el`:

```
$ ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -  9292 -      ?        00:00:00 systemd
4 S     0    19     1  0  80   0 -  8817 -      ?        00:00:00 systemd-journal
4 S   104    61     1  0  80   0 - 64097 -      ?        00:00:00 rsyslogd
4 S     0    63     1  0  80   0 -  7244 -      ?        00:00:00 cron
1 S     0   126     1  0  80   0 -  4031 -      ?        00:00:00 dhclient
4 S     0   154     1  0  80   0 -  3937 -      pts/0    00:00:00 agetty
4 S     0   155     1  0  80   0 -  3937 -      pts/1    00:00:00 agetty
4 S     0   156     1  0  80   0 -  3937 -      pts/2    00:00:00 agetty
4 S     0   157     1  0  80   0 -  3937 -      pts/3    00:00:00 agetty
4 S     0   158     1  0  80   0 -  3937 -      console  00:00:00 agetty
4 S     0   160     1  0  80   0 - 16377 -      ?        00:00:00 sshd
4 S     0   280     0  0  80   0 -  5301 -      ?        00:00:00 bash
0 R     0   392   280  0  80   0 -  7221 -      ?        00:00:00 ps
```

A coluna `PRI` indica a prioridade estática atribuída pelo kernel. Observe, no entanto, que o valor de prioridade exibido por `ps` difere do obtido no exemplo anterior. Por razões históricas, as prioridades exibidas por `ps` variam de -40 a 99 por padrão, de modo que a prioridade real é obtida adicionando 40 a esse valor (no caso, 80 + 40 = 120).

Também é possível monitorar continuamente os processos atualmente gerenciados pelo kernel do Linux com o programa `top`. Como `ps`, `top` também exibe o valor de prioridade de forma diferente. Para facilitar a identificação de processos em tempo real, `top` subtrai 100 do valor da prioridade,

tornando todas as prioridades em tempo real negativas, com um número negativo ou *rt* que as identifica. Portanto, as prioridades normais exibidas por `top` variam de 0 a 39.

NOTE

Para obter mais detalhes do comando `ps`, podemos usar opções adicionais. Compare a saída deste comando com a de nosso exemplo anterior:

```
$ ps -e -o user,uid,comm,tty,pid,ppid,pri,pmem,pcpu --sort=-pcpu |
head
```

Valor nice

Todo processo normal começa com um valor nice padrão de 0 (prioridade 120). O nome *nice* (agradável, cortês) vem da ideia de que os processos "mais corteses" permitem que outros processos sejam executados antes deles em uma fila de execução particular. Os números nice variam de -20 (menos cortês, prioridade alta) a 19 (mais cortês, prioridade baixa). O Linux também permite atribuir diferentes valores nice a threads do mesmo processo. A coluna `NI` na saída de `ps` indica o número *nice*.

Apenas o usuário root pode diminuir o valor nice de um processo para menos de zero. É possível iniciar um processo com uma prioridade não padrão com o comando `nice`. Por padrão, `nice` muda o valor nice para 10, mas esse valor pode ser especificado com a opção `-n`:

```
$ nice -n 15 tar czf home_backup.tar.gz /home
```

Neste exemplo, o comando `tar` é executado com um valor nice de 15. O comando `renice` pode ser usado para alterar a prioridade de um processo em execução. A opção `-p` indica o número PID do processo alvo. Por exemplo:

```
# renice -10 -p 2164
2164 (process ID) old priority 0, new priority -10
```

As opções `-g` e `-u` são usadas para modificar todos os processos de um determinado grupo ou usuário, respectivamente. Com `renice +5 -g users`, o valor nice dos processos pertencentes a usuários do grupo *users* será aumentado em cinco.

Além de `renice`, a prioridade dos processos pode ser modificada com outros programas, como `top`. No alto da tela principal, o valor nice de um processo pode ser modificado pressionando `r` e, em seguida, o número PID do processo:

```
top - 11:55:21 up 23:38,  1 user,  load average: 0,10, 0,04, 0,05
Tasks:  20 total,   1 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,5 us,  0,3 sy,  0,0 ni, 99,0 id,  0,0 wa,  0,2 hi,  0,0 si,  0,0 st
KiB Mem :  4035808 total,   774700 free,  1612600 used,  1648508 buff/cache
KiB Swap:  7999828 total,  7738780 free,   261048 used.  2006688 avail Mem
PID to renice [default pid = 1]
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0   74232   7904   6416 S 0,000 0,196   0:00.12 systemd
   15 root      20   0   67436   6144   5568 S 0,000 0,152   0:00.03 systemd-
journal
   21 root      20   0   61552   5628   5000 S 0,000 0,139   0:00.01 systemd-logind
   22 message+  20   0   43540   4072   3620 S 0,000 0,101   0:00.03 dbus-daemon
   23 root      20   0   45652   6204   4992 S 0,000 0,154   0:00.06 wickedd-dhcp4
   24 root      20   0   45648   6276   5068 S 0,000 0,156   0:00.06 wickedd-auto4
   25 root      20   0   45648   6272   5060 S 0,000 0,155   0:00.06 wickedd-dhcp6
```

A mensagem `PID to renice [default pid = 1]` aparece, com o primeiro processo listado selecionado por padrão. Para alterar a prioridade de outro processo, digite o PID dele e pressione Enter. A mensagem `Renice PID 1 to value` será exibida (com o número PID solicitado) e um novo valor nice poderá ser atribuído.

Exercícios Guiados

1. Em um sistema multitarefa preventivo, o que acontece quando um processo de prioridade mais baixa está ocupando o processador e um processo de prioridade mais alta é posto na fila para ser executado?

2. Considere a seguinte tela de `top`:

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 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
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06
ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17
migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79
migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01
ksoftirqd/1
```

   Quais PIDs têm prioridades em tempo real?

3. Considere a seguinte listagem de `ps -el`:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
```

```
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

Qual PID tem a prioridade mais alta?

4. Depois de tentar alterar o valor nice de um processo com `renice`, acontece o erro a seguir:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Qual a causa provável desse erro?

Exercícios Exploratórios

1. A alteração da prioridades dos processos geralmente é necessária quando um processo está ocupando muito tempo da CPU. Usando `ps` com opções padrão para imprimir todos os processos do sistema em formato longo, qual sinalizador de `--sort` permite classificar os processos por utilização da CPU, em ordem crescente?

2. O comando `schedtool` pode definir todos os parâmetros de agendamento da CPU de que o Linux é capaz ou exibir informações para determinados processos. Como ele pode ser usado para exibir os parâmetros de agendamento do processo `1750`? Além disso, como `schedtool` pode ser usado para alterar o processo 1750 para tempo real com prioridade -90 (conforme exibido por `top`)?

Resumo

Esta lição trata de como o Linux compartilha o tempo da CPU entre os processos gerenciados. Para garantir o melhor desempenho, os processos mais críticos devem ter prioridade sobre os menos críticos. A lição explica as seguintes etapas:

- Conceitos básicos sobre sistemas de multiprocessamento.
- O que é um agendador de processos e como o Linux o implementa.
- O que são as prioridades no Linux, números nice e sua finalidade.
- Como ler e interpretar prioridades de processos no Linux.
- Como alterar a prioridade de um processo antes e durante sua execução.

Respostas aos Exercícios Guiados

1. Em um sistema multitarefa preventivo, o que acontece quando um processo de prioridade mais baixa está ocupando o processador e um processo de prioridade mais alta é posto na fila para ser executado?

   O processo de prioridade mais baixa é pausado e o de prioridade mais alta é executado em seu lugar.

2. Considere a seguinte tela de `top`:

```
top - 08:43:14 up 23 days, 12:29,  5 users,  load average: 0,13, 0,18, 0,21
Tasks: 240 total,   2 running, 238 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,4 us,  0,4 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7726,4 total,    590,9 free,   1600,8 used,   5534,7 buff/cache
MiB Swap:  30517,0 total,  30462,5 free,     54,5 used.   5769,4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0  171420  10668   7612 S   0,0   0,1   9:59.15 systemd
    2 root      20   0       0      0      0 S   0,0   0,0   0:02.76 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
    8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00
mm_percpu_wq
    9 root      20   0       0      0      0 S   0,0   0,0   0:49.06
ksoftirqd/0
   10 root      20   0       0      0      0 I   0,0   0,0  18:24.20 rcu_sched
   11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_bh
   12 root      rt   0       0      0      0 S   0,0   0,0   0:08.17
migration/0
   14 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0,0   0,0   0:11.79
migration/1
   17 root      20   0       0      0      0 S   0,0   0,0   0:26.01
ksoftirqd/1
```

   Quais PIDs têm prioridades em tempo real?

   PIDs 12 and 16.

3. Considere a seguinte listagem de `ps -el`:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 42855 -      ?        00:09:59 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:02 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:49 ksoftirqd/0
1 I     0    10     2  0  80   0 -     0 -      ?        00:18:26 rcu_sched
1 I     0    11     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
1 S     0    12     2  0 -40   - -     0 -      ?        00:00:08 migration/0
1 S     0    14     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/0
5 S     0    15     2  0  80   0 -     0 -      ?        00:00:00 cpuhp/1
```

Qual PID tem a prioridade mais alta?

O PID 12.

4. Depois de tentar alterar o valor nice de um processo com `renice`, acontece o erro a seguir:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Qual a causa provável desse erro?

Apenas o usuário root pode definir os números nice para menos de zero.

Respostas aos Exercícios Exploratórios

1. A alteração da prioridades dos processos geralmente é necessária quando um processo está ocupando muito tempo da CPU. Usando `ps` com opções padrão para imprimir todos os processos do sistema em formato longo, qual sinalizador de `--sort` permite classificar os processos por utilização da CPU, em ordem crescente?

```
$ ps -el --sort=pcpu
```

2. O comando `schedtool` pode definir todos os parâmetros de agendamento da CPU de que o Linux é capaz ou exibir informações para determinados processos. Como ele pode ser usado para exibir os parâmetros de agendamento do processo `1750`? Além disso, como `schedtool` pode ser usado para alterar o processo 1750 para tempo real com prioridade -90 (conforme exibido por `top`)?

```
$ schedtool 1750
```

```
$ schedtool -R -p 89 1750
```