《Linux 操作系统》第五讲：

进程管理

学时计划：4 学时 理论，0 学时 实验

（无实验教学内容）

教学大纲：

1、进程

2、进程的监控

3、进程管理

4、/PROC 文件系统

5、讨论与思考

Linux 的进程管理与 UNIX 操作系统有着很多共同点，但也有一些独特的特性。Linux 是一种动态系统，能够适应不断变化的计算需求。Linux 计算需求的表现是以进程的通用抽象为中心的。进程可以是短期的，也可以是长期的，因此对进行及其调度进行一般管理就显得极为重要。

本讲介绍 Linux 进程的基本概念和生命周期，并重点介绍 Linux 的进程查看、管理和调度的工具的使用方法。

一、进程

1.1 进程的概念

Linux 是一个多用户多任务的操作系统。

多用户是指多个用户可以在同一时间使用同一个 linux 系统；多任务是指在 Linux 下可以同时执行多个任务，更详细的说，linux 采用了分时管理的方法，所有的任务都放在一个队列中，操作系统根据每个任务的优先级为每个任务分配合适的时间片，每个时间片很短，用户根本感觉不到是多个任务在运行，从而使所有的任务共同分享系统资源，因此 Linux 可以在一个任务还未执行完时，暂时挂起此任务，又去执行另一个任务，过一段时间以后再回来处理这个任务，直到这个任务完成，才从任务队列中去除。

上述的描述是在一台计算机上只有一颗 CPU，并且该 CPU 只有一个核心的情形，在这种环境下，虽然系统可以运行多个任务，但是在某一个时间点，CPU 只能执行一个进程。但是如果一台计算机有多颗 CPU，每颗 CPU 有多个核心的情形下，多任务的处理方式是不同的，因此在某个时间点上，可以有多个进程同时运行。

进程的的基本定义是：在自身的虚拟地址空间运行的一个独立的程序，从操作系统的角度来看，所有在系统上运行的东西，都可以称为一个进程。

需要注意的是程序和进程是有区别的，进程虽然有程序产生，但是它并不是程序，程序是一个进程指令的集合，它可以启用一个或多个进程，同时，程序只占用磁盘空间，而不占用系统运行资源，而进程仅占用系统内存空间，是动态的、可变的，关闭进程，占用的内存资源随之释放。

例如，用户在 Linux 上打开一个文件、就会产生一个打开文件的进程，关闭文件，进程也随机关闭。如果在系统上启动一个服务，例如启动 tomcat 服务，就会产生一个对应的 java 的进程。而如果启动 apache 服务，就会产生多个 httpd 进程。

关于进程的更详细的介绍，请参阅《操作系统原理》。

1.2 进程的分类

按照进程的功能和运行的程序分类，进程可划分为两大类：

系统进程：可以执行内存资源分配和进程切换等管理工作。该进程的运行不受用户的干预，即使是 root 用户也不能干预系统进程的运行。

用户进程：通过执行用户程序、应用程序或内核之外的系统程序而产生的进程，此类进程可以在用户的控制下运行或关闭。

用户进程又可以细分为交互进程、批处理进程和守护进程三类。

交互进程：由一个 shell 终端启动的进程，在执行过程中，需要与用户进行交互操作，可以运行于前台，也可以运行在后台。

批处理进程：该进程是一个进程集合，负责按顺序启动其他的进程。

守护进程：守护进程是一直运行的一种进程，经常在 Linux 系统启动时启动，在系统关闭时终止。它们独立于控制终端并且周期性的执行某种任务或等待处理某些发生的事件。例如 httpd 进程，一直处于运行状态，等待用户的访问。

1.3 进程的属性

1.3.1 进程的状态

一个进程在其生存期内，可处于一组不同的状态下，称为进程状态。进程状态保存在进程任务结构的 state 字段中。

当进程正在等待系统中的资源而处于等待状态时，则称其处于睡眠等待状态。在 Linux 系统中，睡眠等待状态分为可中断的和不可中断的等待状态。

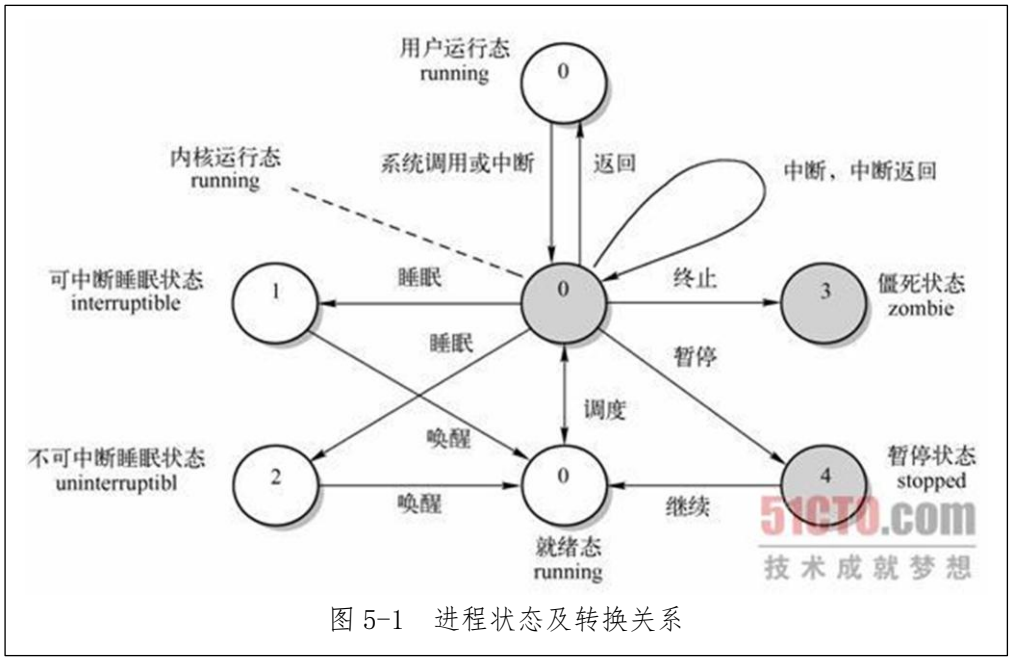

图 5-1　进程状态及转换关系

进程的状态有五种情况，具体如下。

（1）运行状态（TASK_RUNNING）。

当进程正在被 CPU 执行，或已经准备就绪随时可由调度程序执行，则称该进程为处于运行状态(running)。若此时进程没有被 CPU 执行，则称其处于就绪运行状态。见图 5-1 中 3 个标号为 0 的状态。进程可以在内核态运行，也可以在用户态运行。当一个进程在内核代码中运行时，称其处于内核运行态，或简称为内核态；当一个进程正在执行

用户自己的代码时，称其为处于用户运行态（用户态）。当系统资源已经可用时,进程就被唤醒而进入准备运行状态,该状态称为就绪态。这些状态在内核中表示方法相同,都被称为处于 TASK_RUNNING 状态。当一个新进程刚被创建出后就处于本状态中。

（2）可中断睡眠状态（TASK_INTERRUPTIBLE）。

当进程处于可中断等待（睡眠）状态时，系统不会调度该进程执行。当系统产生一个中断或者释放了进程正在等待的资源，或者进程收到一个信号，都可以唤醒进程转换到就绪状态（即可运行状态）。

（3）不可中断睡眠状态（TASK_UNINTERRUPTIBLE）。

除了不会因为收到信号而被唤醒,该状态与可中断睡眠状态类似。但处于该状态的进程只有被使用 wake_up()函数明确唤醒时才能转换到可运行的就绪状态。该状态通常在进程需要不受干扰地等待或者所等待事件会很快发生时使用。

（4）暂停状态（TASK_STOPPED）。

当进程收到信号 SIGSTOP、SIGTSTP、SIGTTIN 或 SIGTTOU 时就会进入暂停状态。可向其发送 SIGCONT 信号让进程转换到可运行状态。进程在调试期间接收到任何信号均会进入该状态。

（5）僵死状态（TASK_ZOMBIE）。

当进程已停止运行，但其父进程还没有调用 wait()询问其状态时，则称该进程处于僵死状态。为了让父进程能够获取其停止运行的信息，此时子进程的任务数据结构信息还需要保留着。一旦父进程调用 wait()取得了子进程的信息，则处于该状态进程的任务数据结构就会被释放。

当一个进程的运行时间片用完,系统就会使用调度程序强制切换到其他的进程去执行。另外，如果进程在内核态执行时需要等待系统的某个资源，此时该进程就会调用 sleep_on() 或 interruptible_sleep_on()自愿地放弃 CPU 的使用权,而让调度程序去执行其他进程。进程则进入睡眠状态（TASK_UNINTERRUPTIBLE 或 TASK_INTERRUPTIBLE）。

只有当进程从"内核运行态"转移到"睡眠状态"时,内核才会进行进程切换操作。在内核态下运行的进程不能被其他进程抢占，而且一

个进程不能改变另一个进程的状态。为了避免进程切换时造成内核数据错误，内核在执行临界区代码时会禁止一切中断。

1.3.2 父进程和子进程

在 Linux 系统中，进程 ID（用 PID 表示）是区分不同进程的唯一标识，它们的大小是有限制的，最大 ID 为 32768，用 UID 和 GID 分别表示启动这个进程的用户和用户组。所有的进程都是 PID 为 1 的 init 进程的后代，内核在系统启动的最后阶段启动 init 进程，因而，这个进程是 Linux 下所有进程的父进程，用 PPID 表示父进程。

使用 ps 命令查看 mysql 的进程信息，具体如下。

```
zhaodongfeng@TeachServer:~$ ps aux|grep mysql
mysql        710  0.0  2.6 176972 27484 ?        Ssl  Oct23   4:17 /usr/sbin/mysqld
1000        4980  0.0  0.1   7928  1036 pts/0    S+   23:36   0:00 grep --color=auto mysql
```

相对于父进程，就存在子进程，一般每个进程都必须有一个父进程，父进程与子进程之间是管理与被管理的关系，当父进程停止时，子进程也随之消失，但是子进程关闭，父进程不一定终止。

如果父进程在子进程退出之前就退出，则所有子进程就变成的一个孤儿进程，如果没有相应的处理机制的话，这些孤儿进程就会一直处于僵死状态，资源无法释放，此时解决的办法是在启动的进程内找一个进程作为这些孤儿进程的父进程，或者直接让 init 进程作为它们的父进程，进而释放孤儿进程占用的资源。

二、进程的监控

不管是 Linux 系统管理员还是程序开发人员，在工作中都不可避免的需要查看系统进程的运行情况，以观测系统的运行情况和进行系统维护操作。

2.1 查看系统进程

2.1.1ps

ps 是最常用的监视进程的命令。ps 命令给出了有关进程的所有有用信息。由于 UNIX 和 Linux 系统的历史比较复杂，发行的版本比较混乱，所以 ps 的用法很多，并且比较混乱。

ps 提供了进程的一次性的查看，它所提供的查看结果并不是动态连续的，而是命令执行时的进程情况。

命令：

```
shell>ps [参数]
```

命令参数：

```
l   长格式输出；
u   按用户名和启动时间的顺序来显示进程；
j   用任务格式来显示进程；
f   用树形格式来显示进程；
a   显示所有用户的所有进程（包括其它用户）；
x   显示无控制终端的进程；
r   显示运行中的进程；
ww  避免详细参数被截断；
```

命令说明：

（1）ps 命令的参数需要以短横杠线开头。

（2）ps 命令的参数可组合使用，例如 ps aux 或 ps lax 等。

（3）ps 命令使用 aux 和 lax 参数时可以显示当前系统上运行的所有进程的信息，显示信息的字段的具体含义见表 5-1 所示。

如下表所示。

表 5-1 PS 命令查看进程信息的字段含义一览表

序号	字段	含义
1	USER	进程的属主。
2	PID	进程的 ID。
3	PPID	父进程的 ID。
4	%CPU	进程占用的 CPU 百分比。
5	%MEM	占用内存的百分比。
6	NI	进程的 NICE 值，数值大，表示较少占用 CPU 时间，优先级低。
7	VSZ	进程占用的虚拟内存的大小。
8	RSS	驻留内存中页的数量（也是管理内存的单位，通常为 4K）。
9	TTY	进程所在终端的 ID 号。
10	WCHAN	正在等待的进程资源。
11	START	启动进程的时间。

12	TIME	进程已经消耗 CPU 的时间。
13	COMMAND	命令的名称和参数。
14	STAT	**进程状态，常用字母代表具体的状态。具体内容如下。** D 睡眠中，等待 I/O 设备。Uninterruptible sleep (usually IO) R 正在运行中的进程，可运行的进程。 S 处于休眠状态，可以被唤醒。 T 停止或被追踪的进程。 W 进入内存交换。（从内核 2.6 开始无效） X 死掉的进程。（很难见到 X 进程状态） Z 僵尸进程。（已经结束却没有释放系统资源的进程） **在进程状态中，常用的附加标志的具体含义如下。** < 优先级高于普通优先级的进程。 N 优先级低于普通优先级的进程。 L 有些页被锁进内存。 s 进程的领导者，也就是说在该进程之下有子进程。 l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)。 + 位于后台的进程组。

2.1.2ps aux

ps 命令非常复杂,也显得混乱,但是通常不需要考虑 ps 的命令，只需要使用固定的组合就可以了。最常用的 ps 组合是 ps aux。

命令：

```
shell>ps aux
```

命令说明：

（1）ps aux 命令显示了系统上运行的所有进程的信息，这些信息的字段代表的具体含义参考表 5-1。

（2）ps aux 可以和 more 结合使用，实现分页查看进程信息。具体命令为：

```
shell>ps aux |more
```

（3）ps aux 命令可以将执行的结果存入文本文件，具体的命令为：

```
shell>ps aux > /home/zhaodongfeng/ps-20111030.txt
shell>cat /home/zhaodongfeng/ps-20111030.txt
shell>more /home/zhaodongfeng/ps-20111030.txt
```

（4）ps aux 和 grep 结合，可以方便的提取指定程序的进程。例如查看 mysqld 和 apache 进程的具体命令为：

```
shell>ps aux |grep mysqld
shell>ps aux |grep apache2
```

2.1.3ps lax

ps 的另外一个组合 ps lax 也经常使用，通过 ps lax 主要可以查看父进程 ID（PPID）和进程优先级（NI）。ps lax 由于不显示进程属主的用户名，而是显示了 UID，因此命令执行的性能要比 ps aux 更快。ps aux 在执行时，首先读取进程属主的 UID，之后再讲 UID 转换为用户名后才输出。

ps lax 和 ps aux 使用的方法基本一致，在此不再累述。

2.2 即时跟踪进程信息

ps 命令可以一次性的给出当前系统中进程信息的快照，但是这样的信息往往缺乏时效性。当管理员需要实时查看系统的进程信息时，就显得不是很方便。

和 ps 不一致的是，top 是动态监视系统任务的工具，top 输出的结果是连续的。

命令：

```
shell>top [参数]
```

命令参数：

```
-b      以批量模式运行，但不能接受命令行输入。
-c      显示命令行，而不仅仅是命令名。
-d N    显示两次刷新时间的间隔，比如 -d 5，表示两次刷新间隔为 5 秒。
-i      禁止显示空闲进程或僵尸进程。
-n NUM  显示更新次数，然后退出。比如-n 5，表示 top 更新 5 次数据就退出。
-p PID      仅监视指定进程的 ID，PID 是一个数值。
-q  不经任何延时就刷新。
-s  安全模式运行，禁用一些效互指令。
-S  累积模式，输出每个进程的总的 CPU 时间，包括已死的子进程。
```

命令说明：

（1）top 命令执行显示的信息会占满全屏幕，默认情况下是 10s

跟新一次。

（2）在 top 命令运行情况下，可以使用命令进行操作。在 top 命令工作区内，可以使用的命令如下所示。

space	立即更新；
c	切换到命令名显示，或显示整个命令（包括参数）。
f,F	增加显示字段，或删除显示字段。
h,?	显示有关安全模式及累积模式的帮助信息。
k	提示输入要杀死的进程 ID，目的是用来杀死该进程（默人信号为 15）
i	禁止空闲进程和僵尸进程。
l	切换到显法负载平均值和正常运行的时间等信息。
m	切换到内存信息，并以内存占用大小排序。
n	提示显示的进程数，比如输入 3，就在整屏上显示 3 个进程。
o,O	改变显示字段的顺序。
r	把 renice 应用到一个进程，提示输入 PID 和 renice 的值。
s	改变两次刷新时间间隔，以秒为单位。
t	切换到显示进程和 CPU 状态的信息。
A	按进程生命大小进行排序，最新进程显示在最前。
M	按内存占用大小排序，由大到小。
N	以进程 ID 大小排序，由大到小。
P	按 CPU 占用情况排序，由大到小。
S	切换到累积时间模式。
T	按时间／累积时间对任务排序。
W	把当前的配置写到~/.toprc 中。

2.3 查看指定程序的进程信息

pgrep 是通过程序的名字来查询进程的工具，一般是用来判断程序是否正在运行。在服务器的配置和管理中，这个工具常被应用，帮助服务器管理员快速的确定某一服务的状态。

命令：

```
shell>pgrep [参数] 程序名称
```

命令参数：

```
-l  列出程序名和进程 ID。
-o  进程起始的 ID。
-n  进程终止的 ID。
```

命令举例：

```
shell>pgrep apache2
shell>pgrep -l apache2
shell>pgrep -o apache2
shell>pgrep -n apache2
shell>pgrep -lo mysql
shell>pgrep -lon mysql
shell>pgrep -ln mysql
```

2.4 查看指定文件的进程信息

lsof（list open files）是一个列出当前系统打开文件的工具。在 Linux 环境下，任何事物都以文件的形式存在，通过文件不仅仅可以访问常规数据，还可以访问网络连接和硬件。

在终端下输入 lsof 即可显示系统打开的文件，因为 lsof 需要访问核心内存和各种文件，所以 lsof 命令需要 root 权限进行执行时才能够执行全部功能。lsof 命令的显示结果中每行显示一个打开的文件，若不指定条件默认将显示所有进程打开的所有文件。

例如，使用 lsof 查看系统打开的所有文件，结果如下。

```
managebyruanon3-3@HactcmServer3-3:~$ head -n 10 /home/managebyruanon3-3/lsof-20111030.txt
COMMAND   PID     USER   FD      TYPE             DEVICE SIZE/OFF      NODE NAME
init        1     root  cwd       DIR                8,1     4096          2 /
init        1     root  rtd       DIR                8,1     4096          2 /
init        1     root  txt       REG                8,1   146424    4980778 /sbin/init
init        1     root  mem       REG                8,1    51728   62914592 /lib/x86_64-linux-gnu/libnss_files-2.13.so
init        1     root  mem       REG                8,1    47680   62914594 /lib/x86_64-linux-gnu/libnss_nis-2.13.so
init        1     root  mem       REG                8,1    97248   62914589 /lib/x86_64-linux-gnu/libnsl-2.13.so
init        1     root  mem       REG                8,1    35712   62914590 /lib/x86_64-linux-gnu/libnss_compat-2.13.so
init        1     root  mem       REG                8,1  1638120   62914579 /lib/x86_64-linux-gnu/libc-2.13.so
init        1     root  mem       REG                8,1    31752   62914599 /lib/x86_64-linux-gnu/librt-2.13.so
```

lsof 输出各列信息的意义如表 5-2 所示。

表 5-2 lsof 命令查看进程信息的字段含义一览表

序号	字段	含义
1	COMMAND	进程的名称。
2	PID	进程标识符。
3	USER	进程所有者。
4	FD	文件描述符，应用程序通过文件描述符识别该文件。如 cwd、txt 等。
5	TYPE	文件类型，如 DIR、REG 等。
6	DEVICE	指定磁盘的名称。
7	SIZE	文件的大小。
8	NODE	索引节点（文件在磁盘上的标识）。
9	NAME	打开文件的确切名称。

命令：

```
shell>lsof [参数] [文件名]
```

命令参数：

```
filename 显示打开指定文件的所有进程
-a 表示两个参数都必须满足时才显示结果
-c string    显示 COMMAND 列中包含指定字符的进程所有打开的文件
-u username  显示所属 user 进程打开的文件
-g gid 显示归属 gid 的进程情况
+d /DIR/ 显示目录下被进程打开的文件
+D /DIR/ 同上，但是会搜索目录下的所有目录，时间相对较长
-d FD 显示指定文件描述符的进程
-n 不将 IP 转换为 hostname，缺省是不加上-n 参数
-i 用以显示符合条件的进程情况。命令格式是：
        -i[46] [protocol][@hostname|hostaddr][:service|port]
        具体参数含义：
          46 --> IPv4 or IPv6
          protocol --> TCP or UDP
          hostname --> Internet host name
          hostaddr --> IPv4 地址
          service --> /etc/service 中的 service name
          port --> 端口号
```

命令举例：

（1）查看占用指定文件的进程和用户

```
shell>lsof /sbin/init
```

（2）查看占用某个文件目录的进程和用户

```
shell>lsof /proc/
```

（3）查看占用指定端口的进程

```
shell>lsof -i :22
```

（4）查看 root 用户所打开的 txt 类型的文件

```
shell>lsof -a -u root -c .txt
```

（5）查看使用 IPv4 通信的进程

```
shell>lsof -i 4
```

（6）查看使用 IPv6 通信的进程

```
shell>lsof -i 6
```

（7）查看和指定 IP 地址通信的进程

```
shell>lsof -i @211.69.44.2
```

三、进程的管理

在系统运行中，不可避免的会有一些进程发生问题，变成僵死进程，也会因为某种需要终止一些进程的运行。在某些时候，需要提供指定进程运行的优先级，让其能够使用更多的 CPU 的资源。

在 Linux 系统中，提供了一系列对进程管理的工具。

3.1 终止进程

3.1.1kill

kill 命令用来终止一个进程或终止一个正在运行的程序。从本质上讲，kill 命令只是用来向进程发送一个信号，具体发送的信号内容有用户通过 kill 命令的参数来指定。

命令：

```
kill [参数] PID
```

命令参数：

```
-s：指定发送的信号。
-p：模拟发送信号。
-l：指定信号的名称列表。
PID：要中止进程的 ID 号。
Signal：表示信号。
```

命令说明：

（1）kill 命令的工作原理是：向 Linux 系统的内核发送一个系统操作信号和某个程序的进程标识号，然后系统内核就可以对进程标识号指定的进程进行操作。

（2）kill 发送的信号可以通过 kill -l 命令查看。具体如下。

```
$ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL      10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

编号为 1-31 的信号为传统 UNIX 支持的信号，是不可靠信号（非实时的），编号为 32-63 的信号是后来扩充的，称做可靠信号（实时信号）。不可靠信号和可靠信号的区别在于前者不支持排队，可能会造成信号丢失，而后者不会。

常用的信息选项的介绍如表 5-3 所示。

表 5-3 常用 kill 命令信号介绍

信号编号	信号名	描述	默认操作
0	EXIT	程序退出时收到该信号	终止
1	HUP	挂起	终止
2	INT	中断	终止
3	QUIT	退出	终止
9	KILL	杀死	终止
11	SEGV	段错误	终止
15	TERM	软件终止	终止

命令举例：

通过查看进程，关闭 mysqld 进程。

（1）查看 mysqld 的进程信息，获得 PID。

（2）强制关闭 mysqld 进程。

具体命令如下：

```
#查看 mysqld 的进程号。
zhaodongfeng@TeachServer:~$ sudo ps auxf |grep mysqld
1000 10545 0.0 0.0 15576 1060 pts/0 S+ 20:40 0:00 \_ grep --color=auto mysqld
mysql 9639 0.5 0.1 268972 26824 ? Ssl 20:39 0:00 /usr/sbin/mysqld

#通过查看得到 mysqld 的进程号是 9636。
#执行 kill 命令，终止 PID 为 9636 的进程
zhaodongfeng@TeachServer:~$ sudo kill -kill 9639

#重新查看 mysqld 的进程，会发现 mysqld 的进程号为 10555。
#说明 mysqld 的进程已关闭成功，但 mysqld 重新启动了服务，获得新的 PID。
zhaodongfeng@TeachServer:~$ sudo ps auxf |grep mysqld
1000 10628 0.0 0.0 15576 1060 pts/0 S+ 20:40 0:00 \_ grep --color=auto mysqld
mysql 10555 4.6 0.1 268972 26828 ? Ssl 20:40 0:00 /usr/sbin/mysqld
```

3.1.2killall

killall 命令是通过程序的名字，直接杀死所有进程，而不需要查询 PID。

命令：

```
killall 程序名
```

命令举例：

```
#查看 mysqld 的程序名称。
zhaodongfeng@TeachServer:~$ sudo pgrep -l mysqld
10555 mysqld

#使用 killall 直接终止 mysqld 程序的运行。
zhaodongfeng@TeachServer:~$ sudo killall mysqld

#查看 mysqld 的进程信息，看到 mysqld 的 PID 发生了变化。
zhaodongfeng@TeachServer:~$ sudo pgrep -l mysqld
11478 mysqld
```

3.2 调整进程的优先级

在 Linux 操作系统中，进程之间是竞争资源（比如 CPU 和内存的占用）关系。进程间的竞争关系中的优势是通过一个数值来实现的，也就是谦让度。高谦让度表示进程优先级别最低。负值或 0 表示对高优点级，对其它进程不谦让，也就是拥有优先占用系统资源的权利。谦让度的值从-20 到 19。

目前硬件技术发展极速，在大多情况下，不必设置进程的优先级，除非在进程失控而疯狂占用资源的情况下，有可能来设置一下优先级。但是在实际应用中，对于疯狂占用资源的进程，通常终止进程会比调整优先级更实际。

3.2.1nice

nice 命令可以在创建进程时，为进程指定谦让度的值，进程的优先级的值是父进程优先级的值与所指定谦让度的值相加之和。因此，在用 nice 设置程序的优先级时，所指定的数值是进程优先级的增量，并不是进程优先级的绝对值。

命令：

```
nice -n 谦让度相对值 程序名
```

命令说明：

（1）nice 命令可以以不同的谦让度启动程序，也就是说 nice 命令要求程序是在没有启动的情况下，以不同的谦让度启动程序。

（2）nice 命令所指定的谦让度是相对值，最终的谦让度是默认值和指定值之和。

命令举例：

```
#关闭 apache2 服务。
zhaodongfeng@TeachServer:~$ sudo /etc/init.d/apache2 stop
 *Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 211.69.44.22 for ServerName
 ...                                                                                                    waiting
[ OK ]
#增加谦让度 10，开启 apache2 服务。
zhaodongfeng@TeachServer:~$ sudo nice -n 10 /etc/init.d/apache2 start
 * Starting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 211.69.44.22 for
```

```
ServerName

[ OK ]
#查看 apache2 的进程信息，NI 为 10。
zhaodongfeng@TeachServer:~$ ps lax |grep apache2
1     0  4548     1  30  10 158092  9132 poll_s SNs  ?          0:00 /usr/sbin/apache2 -k start
5    33  4553  4548  30  10 158116  5300 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
5    33  4554  4548  30  10 158116  5300 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
5    33  4555  4548  30  10 158116  5300 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
5    33  4556  4548  30  10 158116  5300 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
5    33  4557  4548  30  10 158116  5300 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
0  1000  4559  3866  20   0   7892   872 pipe_w S+   pts/0      0:00 grep --color=auto apache2
#关闭 apache2 服务。
zhaodongfeng@TeachServer:~$ sudo /etc/init.d/apache2 stop
 * Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 211.69.44.22 for ServerName
 ...                                                                                                  waiting
[ OK ]
#增加谦让度-8，启动 apache2 服务。
zhaodongfeng@TeachServer:~$ sudo nice -n -8 /etc/init.d/apache2 start
 * Starting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 211.69.44.22 for ServerName

[ OK ]
#查看 apache2 的进程信息，NI 为-8。
zhaodongfeng@TeachServer:~$ ps lax |grep apache2
1     0  4603     1  12  -8 158092  9136 poll_s S<s  ?          0:00 /usr/sbin/apache2 -k start
5    33  4608  4603  12  -8 158116  5304 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
5    33  4609  4603  12  -8 158116  5304 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
5    33  4610  4603  12  -8 158116  5304 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
5    33  4611  4603  12  -8 158116  5304 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
5    33  4612  4603  12  -8 158116  5304 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
0  1000  5417  3866  20   0   7892   876 pipe_w S+   pts/0      0:00 grep --color=auto apache2
```

3.2.2renice

命令：

```
renice 谦让度绝对值 PID
```

命令说明：

（1）renice 是对正在运行的进程进行谦让度值的变更。

（2）renice 指定的谦让度值是绝对值。

（3）renice 依据进程的 PID 号进行谦让度的变更。

命令举例：

```
#查看 apache2 的进程信息，NI 为 0，PID 为 6892。
zhaodongfeng@TeachServer:~$ ps lax |grep apache2
1     0  6887     1  20   0 158092  9212 poll_s Ss   ?          0:00 /usr/sbin/apache2 -k start
5    33  6892  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6893  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6894  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6895  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6896  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6900  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
0  1000  6916  3866  20   0   7892   876 pipe_w S+   pts/0      0:00 grep --color=auto apache2
#定义 PID 为 6832 的进程的谦让度为 5。
zhaodongfeng@TeachServer:~$ sudo renice -n 5 -p 6892
6892 (process ID) old priority 0, new priority 5
#查看 apache2 的进程信息，NI 为 5，PID 为 6892。
zhaodongfeng@TeachServer:~$ ps lax |grep apache2
1     0  6887     1  20   0 158092  9212 poll_s Ss   ?          0:00 /usr/sbin/apache2 -k start
5    33  6892  6887  25   5 158172  5792 inet_c SN   ?          0:00 /usr/sbin/apache2 -k start
5    33  6893  6887  20   0 158172  5792 poll_s S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6894  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6895  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6896  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6900  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
0  1000  6920  3866  20   0   7892   876 pipe_w S+   pts/0      0:00 grep --color=auto apache2
#定义 PID 为 6832 的进程的谦让度为-10。
zhaodongfeng@TeachServer:~$ sudo renice -n -10 -p 6892
6892 (process ID) old priority 5, new priority -10
#查看 apache2 的进程信息，NI 为-10，PID 为 6892。
zhaodongfeng@TeachServer:~$ ps lax |grep apache2
1     0  6887     1  20   0 158092  9212 poll_s Ss   ?          0:00 /usr/sbin/apache2 -k start
5    33  6892  6887  10 -10 158172  5792 inet_c S<   ?          0:00 /usr/sbin/apache2 -k start
5    33  6893  6887  20   0 158172  5792 poll_s S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6894  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6895  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6896  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
5    33  6900  6887  20   0 158172  5792 inet_c S    ?          0:00 /usr/sbin/apache2 -k start
0  1000  6924  3866  20   0   7892   876 pipe_w S+   pts/0      0:00 grep --color=auto apache2
```

四、/PROC 文件系统

Linux 系统上的/proc 目录是一种文件系统，即 proc 文件系统。与其它常见的文件系统不同的是，/proc 是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息，甚至可以通过更改其中某些文件来改变内核的运行状态。

基于/proc 文件系统如上所述的特殊性，其内的文件也常被称作虚拟文件，并具有一些独特的特点。例如，其中有些文件虽然使用查看命令查看时会返回大量信息，但文件本身的大小却会显示为 0 字节。此外，这些特殊文件中大多数文件的时间及日期属性通常为当前系统时间和日期，这跟它们随时会被刷新（存储于 RAM 中）有关。

为了查看及使用上的方便，这些文件通常会按照相关性进行分类存储于不同的目录甚至子目录中，如/proc/scsi 目录中存储的就是当前系统上所有 SCSI 设备的相关信息，/proc/N 中存储的则是系统当前正在运行的进程的相关信息，其中 N 为正在运行的进程。

大多数虚拟文件可以使用文件查看命令如 cat、more 或者 less 进行查看，有些文件信息表述的内容可以一目了然，但也有文件的信息却不怎么具有可读性。不过，这些可读性较差的文件在使用一些命令如 apm、free、lspci 或 top 查看时却可以有着不错的表现。

4.1/proc 目录下的进程目录

/proc 目录中包含许多以数字命名的子目录，这些数字表示系统当前正在运行进程的进程号，里面包含对应进程相关的多个信息文件。使用 ll /proc/进程号可以查看进程号的详细内容。

举例：

```
zhaodongfeng@TeachServer:~$ ll /proc
total 4
dr-xr-xr-x 79 root          root          0 2011-10-30 09:58 ./
drwxr-xr-x 23 root          root       4096 2011-10-30 08:31 ../
dr-xr-xr-x  8 root          root          0 2011-10-30 09:58 1/
dr-xr-xr-x  8 root          root          0 2011-10-30 09:59 10/
dr-xr-xr-x  8 root          root          0 2011-10-30 10:05 1018/
dr-xr-xr-x  8 root          root          0 2011-10-30 09:59 11/
dr-xr-xr-x  8 root          root          0 2011-10-30 09:59 12/
```

```
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 13/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 14/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 15/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 16/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 18/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 195/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 197/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 2/
dr-xr-xr-x    8 root                    root                    0 2011-10-30 09:59 235/
dr-xr-xr-x    8 root                    root                    0 2011-10-31 09:26 24319/

zhaodongfeng@TeachServer:~$ ll /proc/6900
ls: cannot read symbolic link /proc/6900/cwd: Permission denied
ls: cannot read symbolic link /proc/6900/root: Permission denied
ls: cannot read symbolic link /proc/6900/exe: Permission denied
total 0
dr-xr-xr-x    8 www-data www-data 0 2011-10-30 10:56 ./
dr-xr-xr-x 79 root         root         0 2011-10-30 09:58 ../
dr-xr-xr-x    2 www-data www-data 0 2011-10-31 09:26 attr/
-rw-r--r--    1 root         root         0 2011-10-31 09:26 autogroup
-r--------    1 root         root         0 2011-10-31 09:26 auxv
-r--r--r--    1 root         root         0 2011-10-31 09:26 cgroup
--w-------    1 root         root         0 2011-10-31 09:26 clear_refs
-r--r--r--    1 root         root         0 2011-10-30 10:56 cmdline
-rw-r--r--    1 root         root         0 2011-10-31 09:26 comm
-rw-r--r--    1 root         root         0 2011-10-31 09:26 coredump_filter
-r--r--r--    1 root         root         0 2011-10-31 09:26 cpuset
lrwxrwxrwx    1 root         root         0 2011-10-30 11:09 cwd
-r--------    1 root         root         0 2011-10-31 09:26 environ
lrwxrwxrwx    1 root         root         0 2011-10-30 11:09 exe
dr-x------    2 root         root         0 2011-10-30 11:09 fd/
dr-x------    2 root         root         0 2011-10-31 09:26 fdinfo/
-r--------    1 root         root         0 2011-10-31 09:26 io
-r--r--r--    1 root         root         0 2011-10-31 09:26 latency
-r--r--r--    1 root         root         0 2011-10-31 09:26 limits
-rw-r--r--    1 root         root         0 2011-10-31 09:26 loginuid
-r--r--r--    1 root         root         0 2011-10-30 11:09 maps
-rw-------    1 root         root         0 2011-10-31 09:26 mem
-r--r--r--    1 root         root         0 2011-10-31 09:26 mountinfo
-r--r--r--    1 root         root         0 2011-10-31 09:26 mounts
-r--------    1 root         root         0 2011-10-31 09:26 mountstats
dr-xr-xr-x    5 www-data www-data 0 2011-10-31 09:26 net/
dr-x--x--x    2 root         root         0 2011-10-31 09:26 ns/
-r--r--r--    1 root         root         0 2011-10-31 09:26 numa_maps
-rw-r--r--    1 root         root         0 2011-10-31 09:26 oom_adj
-r--r--r--    1 root         root         0 2011-10-31 09:26 oom_score
```

```
-rw-r--r--  1 root     root     0 2011-10-31 09:26 oom_score_adj
-r--r--r--  1 root     root     0 2011-10-31 09:26 pagemap
-r--r--r--  1 root     root     0 2011-10-31 09:26 personality
lrwxrwxrwx  1 root     root     0 2011-10-30 11:09 root
-rw-r--r--  1 root     root     0 2011-10-31 09:26 sched
-r--r--r--  1 root     root     0 2011-10-31 09:26 schedstat
-r--r--r--  1 root     root     0 2011-10-31 09:26 seccomp_filter
-r--r--r--  1 root     root     0 2011-10-31 09:26 sessionid
-r--r--r--  1 root     root     0 2011-10-31 09:26 smaps
-r--r--r--  1 root     root     0 2011-10-31 09:26 stack
-r--r--r--  1 root     root     0 2011-10-30 10:56 stat
-r--r--r--  1 root     root     0 2011-10-31 09:26 statm
-r--r--r--  1 root     root     0 2011-10-30 10:56 status
-r--r--r--  1 root     root     0 2011-10-31 09:26 syscall
dr-xr-xr-x  3 www-data www-data 0 2011-10-31 09:26 task/
-r--r--r--  1 root     root     0 2011-10-30 10:56 wchan
```

在实例中，出现的目录中比较关键的目录代表的意义如下。

（1）cmdline：启动当前进程的完整命令，但僵尸进程目录中的此文件不包含任何信息。

（2）cwd：指向当前进程运行目录的一个符号链接。

（3）environ：当前进程的环境变量列表，彼此间用空字符（NULL）隔开；变量用大写字母表示，其值用小写字母表示。

（4）exe：指向启动当前进程的可执行文件（完整路径）的符号链接，通过/proc/N/exe 可以启动当前进程的一个拷贝。

（5）fd：这是个目录，包含当前进程打开的每一个文件的文件描述符（file descriptor），这些文件描述符是指向实际文件的一个符号链接。

（6）limits：当前进程所使用的每一个受限资源的软限制、硬限制和管理单元；此文件仅可由实际启动当前进程的 UID 用户读取。

（7）maps：当前进程关联到的每个可执行文件和库文件在内存中的映射区域及其访问权限所组成的列表。

（8）mem：当前进程所占用的内存空间，由 open、read 和 lseek 等系统调用使用，不能被用户读取。

（9）root：指向当前进程运行根目录的符号链接；在 Unix 和 Linux 系统上，通常采用 chroot 命令使每个进程运行于独立的根目

录。

（10）stat：当前进程的状态信息，包含一系统格式化后的数据列，可读性差，通常由 ps 命令使用。

（11）statm：当前进程占用内存的状态信息，通常以“页面”（page）表示。

（12）status：与 stat 所提供信息类似，但可读性较好，如下所示，每行表示一个属性信息；其详细介绍请参见 proc 的 man 手册页。

（13）task：目录文件，包含由当前进程所运行的每一个线程的相关信息，每个线程的相关信息文件均保存在一个由线程号（tid）命名的目录中，这类似于其内容类似于每个进程目录中的内容。

4.2/proc 的常见目录

/proc 目录下的许多目录表示的是系统运行的信息。具体内容如下描述所示。

（1）/proc/apm

高级电源管理（APM）版本信息及电池相关状态信息，通常由 apm 命令使用。

（2）/proc/buddyinfo

用于诊断内存碎片问题的相关信息文件。

（3）/proc/cmdline

在启动时传递至内核的相关参数信息，这些信息通常由 lilo 或 grub 等启动管理工具进行传递。

（4）/proc/cpuinfo

处理器的相关信息的文件。

（5）/proc/crypto

系统上已安装的内核使用的密码算法及每个算法的详细信息列表。

（6）/proc/devices

系统已经加载的所有块设备和字符设备的信息，包含主设备号和设备组（与主设备号对应的设备类型）名。

（7）/proc/diskstats

每块磁盘设备的磁盘 I/O 统计信息列表（内核 2.5.69 以后的版本支持此功能）。

（8）/proc/dma

每个正在使用且注册的 ISA DMA 通道的信息列表。

（9）/proc/execdomains

内核当前支持的执行域(每种操作系统独特“个性”)信息列表。

（10）/proc/fb

帧缓冲设备列表文件，包含帧缓冲设备的设备号和相关驱动信息。

（11）/proc/filesystems

当前被内核支持的文件系统类型列表文件，被标示为 nodev 的文件系统表示不需要块设备的支持；通常 mount 一个设备时，如果没有指定文件系统类型将通过此文件来决定其所需文件系统的类型。

（12）/proc/interrupts

X86 或 X86_64 体系架构系统上每个 IRQ 相关的中断号列表；多路处理器平台上每个 CPU 对于每个 I/O 设备均有自己的中断号。

（13）/proc/iomem

每个物理设备上的记忆体（RAM 或者 ROM）在系统内存中的映射信息。

（14）/proc/ioports

当前正在使用且已经注册过的与物理设备进行通讯的输入-输出端口范围信息列表。

（15）/proc/kallsyms

模块管理工具用来动态链接或绑定可装载模块的符号定义，由内核输出（内核 2.5.71 以后的版本支持此功能），通常这个文件中的信息量相当大。

（16）/proc/kcore

系统使用的物理内存，以 ELF 核心文件（core file）格式存储，其文件大小为已使用的物理内存（RAM）加上 4KB；这个文件用来检查内核数据结构的当前状态，因此，通常由 GBD 通常调试工具使用，但不能使用文件查看命令打开此文件。

（17）/proc/kmsg

此文件用来保存由内核输出的信息，通常由/sbin/klogd 或/bin/dmsg 等程序使用，不要试图使用查看命令打开此文件。

（18）/proc/loadavg

保存关于CPU 和磁盘I/O 的负载平均值，其前三列分别表示每1 秒钟、每5 秒钟及每15 秒的负载平均值，类似于uptime 命令输出的相关信息；第四列是由斜线隔开的两个数值，前者表示当前正由内核调度的实体（进程和线程）的数目，后者表示系统当前存活的内核调度实体的数目；第五列表示此文件被查看前最近一个由内核创建的进程的PID。

（19）/proc/locks

保存当前由内核锁定的文件的相关信息，包含内核内部的调试数据；每个锁定占据一行，且具有一个惟一的编号；如下输出信息中每行的第二列表示当前锁定使用的锁定类别，POSIX 表示目前较新类型的文件锁，由 lockf 系统调用产生，FLOCK 是传统的 UNIX 文件锁，由 flock 系统调用产生；第三列也通常由两种类型，ADVISORY 表示不允许其他用户锁定此文件，但允许读取，MANDATORY 表示此文件锁定期间不允许其他用户任何形式的访问；

（20）/proc/mdstat

保存RAID 相关的多块磁盘的当前状态信息，在没有使用RAID 机器上，其显示为<none>。

（21）/proc/meminfo

系统中关于当前内存的利用状况等的信息，常由 free 命令使用；可以使用文件查看命令直接读取此文件，其内容显示为两列，前者为统计属性，后者为对应的值。

（22）/proc/mounts

在内核 2.4.29 版本以前，此文件的内容为系统当前挂载的所有文件系统，在 2.4.19 以后的内核中引进了每个进程使用独立挂载名称空间的方式，此文件则随之变成了指向/proc/self/mounts（每个进程自身挂载名称空间中的所有挂载点列表）文件的符号链接。/proc/self 是一个独特的目录。

（23）/proc/modules

当前装入内核的所有模块名称列表，可以由 lsmod 命令使用，也可以直接查看；如下所示，其中第一列表示模块名，第二列表示此模块占用内存空间大小，第三列表示此模块有多少实例被装入，第四列表示此模块依赖于其它哪些模块，第五列表示此模块的装载状态（Live：已经装入；Loading：正在装入；Unloading：正在卸载），第六列表示此模块在内核内存（kernel memory）中的偏移量。

（24）/proc/partitions

块设备每个分区的主设备号（major）和次设备号（minor）等信息，同时包括每个分区所包含的块（block）数目。

（25）/proc/pci

内核初始化时发现的所有 PCI 设备及其配置信息列表，其配置信息多为某 PCI 设备相关 IRQ 信息，可读性不高，可以用“/sbin/lspci -vb”命令获得较易理解的相关信息；在 2.6 内核以后，此文件已为/proc/bus/pci 目录及其下的文件代替。

（26）/proc/slabinfo

在内核中频繁使用的对象（如 inode、dentry 等）都有自己的 cache，即 slab pool，而/proc/slabinfo 文件列出了这些对象相关 slap 的信息；详情可以参见内核文档中 slapinfo 的手册页。

（27）/proc/stat

实时追踪自系统上次启动以来的多种统计信息，其中：

“cpu”行后的八个值分别表示以 1/100（jiffies）秒为单位的统计值(包括系统运行于用户模式、低优先级用户模式，运系统模式、空闲模式、I/O 等待模式的时间等)；

“intr”行给出中断的信息，第一个为自系统启动以来，发生的所有的中断的次数；然后每个数对应一个特定的中断自系统启动以来所发生的次数；

“ctxt”给出了自系统启动以来 CPU 发生的上下文交换的次数。

“btime”给出了从系统启动到现在为止的时间，单位为秒；

“processes (total_forks)” 自系统启动以来所创建的任务的个数目；

“procs_running”：当前运行队列的任务的数目；

“procs_blocked”：当前被阻塞的任务的数目；

（28）/proc/swaps

当前系统上的交换分区及其空间利用信息，如果有多个交换分区的话，则会每个交换分区的信息分别存储于/proc/swap 目录中的单独文件中，而其优先级数字越低，被使用到的可能性越大；下面是作者系统中只有一个交换分区时的输出信息。

（29）/proc/uptime

系统上次启动以来的运行时间，如下所示，其第一个数字表示系统运行时间，第二个数字表示系统空闲时间，单位是秒。

（30）/proc/version

当前系统运行的内核版本号。

（31）/proc/vmstat

当前系统虚拟内存的多种统计数据，信息量可能会比较大，这因系统而有所不同，可读性较好（2.6 以后的内核支持此文件）。

（32）/proc/zoneinfo

内存区域（zone）的详细信息列表，信息量较大。

4.3/proc/sys 目录

与/proc 下其它文件的“只读”属性不同的是，管理员可对/proc/sys 子目录中的许多文件内容进行修改以更改内核的运行特性，事先可以使用“ls -l”命令查看某文件是否“可写入”。需要注意的是，即使文件可写，其一般也不可以使用编辑器进行编辑。

（1）/proc/sys/debug 子目录

此目录通常是一空目录。

（2）/proc/sys/dev 子目录

为系统上特殊设备提供参数信息文件的目录，其不同设备的信息文件分别存储于不同的子目录中，如大多数系统上都会具有的/proc/sys/dev /cdrom 和/proc/sys/dev/raid（如果内核编译时开启了支持 raid 的功能)目录,其内存储的通常是系统上 cdrom 和 raid 的相关参数信息文件。

五、讨论与思考

5.1 进程和线程

（1）进程和线程有什么关系？

（2）如何查看线程的信息？