第 8 章　系统监测与维护

系统管理员的重要工作之一是尽可能提高系统的性能，最大化资源的使用效率并及时对系统进行更新和维护。Linux 系统中提供了一些系统资源监测与维护的命令和工具，综合利用这些工具可以有效提高系统的运行效率。

8.1　监测系统资源和性能

系统资源和性能的监控主要涉及对 CPU 使用率、内存使用率（涉及 RAM 和交换空间）、磁盘空间，以及系统负载的查询与检测。

8.1.1　使用 proc 文件系统查看系统内核信息

proc 不是一个真实的文件系统，不占用外存空间，只是以文件的方式为访问 Linux 内核数据提供接口。用户和应用程序可以通过查看/proc 得到系统的运行信息，并可以改变内核的某些参数。由于系统的信息总是动态变化的，所以用户或应用程序读取 proc 文件时所获得的数据也是瞬时的。许多应用程序依靠 proc 来访问 Linux 内核信息。查看/proc 目录如下：

```
# cd  /proc
# ls
1       1939  2230  307   3287  3377  7             fs           net
10      1952  2231  3072  3289  3379  723           ide          partitions
12      1964  2252  3073  3291  3393  74            interrupts   schedstat
1225    2     2271  3074  3298  3395  78            iomem        scsi
13      2004  2283  3075  3299  3397  80            ioports      self
131     2024  2284  3076  3301  3408  9             irq          slabinfo
132     2043  2295  3077  3313  3420  acpi          kallsyms     stat
133     2062  2296  3078  3317  3428  asound        kcore        swaps
134     2073  2302  312   3319  3430  buddyinfo     keys         sys
1770    2078  2305  317   3323  3431  bus           key-users    sysrq-trigger
1782    2090  2315  3179  3327  3442  cmdline       kmsg         sysvipc
1784    2104  2559  3183  3333  3444  cpuinfo       loadavg      tty
1798    2116  2564  3186  3336  3445  crypto        locks        uptime
1801    2129  2565  320   3343  3520  devices       mdstat       version
1828    2149  2567  3211  3350  353   diskstats     meminfo      vmcore
1841    2158  2569  3240  3352  387   dma           misc         vmstat
1854    2170  276   3274  3354  4     driver        modules      xen
1875    2181  2938  3277  3366  5     execdomains   mounts       zoneinfo
1908    2216  3     3278  3368  5628  fb            mpt
1927    2227  3060  3285  3371  6     filesystems   mtrr
```

1. 查看进程信息

在/proc 目录中，每一个以数字命名的子目录对应系统中运行的一个进程，该数字即为进程的 PID 号。数字目录中存放了该进程的运行信息，可以通过相应的命令进行查询。例如查询当然系统中运行的 vi 进程的相关信息：

```
# ps aux | grep vi
root      5716  0.0  0.4   4956  1012 pts/1    T    11:55   0:00 vi        //5716 为 vi 的进程号
root      5778  0.0  0.2   4136   668 pts/1    R+   12:01   0:00 grep vi
# ls      //查看 proc 目录下是否已存在 5716 目录
1     1939  2230  307   3287  3377  5717          fb            mpt
10    1952  2231  3072  3289  3379  6             filesystems   mtrr
12    1964  2252  3073  3291  3393  7             fs            net
1225  2     2271  3074  3298  3395  723           ide           partitions
13    2004  2283  3075  3299  3397  74            interrupts    schedstat
131   2024  2284  3076  3301  3408  78            iomem           scsi
132   2043  2295  3077  3313  3420  80            ioports       self
133   2062  2296  3078  3317  3428  9             irq           slabinfo
134   2073  2302  312   3319  3430  acpi          kallsyms      stat
1770  2078  2305  317   3323  3431  asound         kcore          swaps
1782  2090  2315  3179  3327  3442  buddyinfo     keys           sys
1784  2104  2559  3183  3333  3444  bus            key-users      sysrq-trigger
1798  2116  2564  3186  3336  3445  cmdline        kmsg           sysvipc
1801  2129  2565  320   3343  3520  cpuinfo        loadavg        tty
1828  2149  2567  3211  3350  353   crypto         locks          uptime
1841  2158  2569  3240  3352  387   devices        mdstat         version
1854  2170  276   3274  3354  4     diskstats      meminfo        vmcore
1875  2181  2938  3277  3366  5     dma            misc           vmstat
1908  2216  3     3278  3368  5685  driver         modules        xen
1927  2227  3060  3285  3371  5716  execdomains    mounts         zoneinfo
# ls   -l   /proc/5716/exe              //查询 vi 所执行的程序
lrwxrwxrwx 1 root root 0 07-25 11:57 /proc/5716/exe -> /bin/vi
# ls   -l   /proc/5716/cwd              //查询 vi 的当前目录
lrwxrwxrwx 1 root root 0 07-25 11:57 /proc/5716/cwd -> /proc
# ls   -l   /proc/5716/environ          //查询 vi 的运行环境
-r-------- 1 root root 0 07-25 11:57 /proc/5716/environ
```

2. 查看 CPU 信息

可以通过查看 cupinfo 文件获得处理器的详细信息；通过 interrupts 文件可以查看当前系统使用的中断号；通过 uptime 文件可以查看系统运行的时间；通过 filesystems 文件，可以查看当前系统支持的文件系统类型。例如，查看系统内存使用情况，可以使用下面的命令：

```
# cat   meminfo
MemTotal:       252444 kB              //总内存 256M
MemFree:          6428 kB
Buffers:          1828 kB
Cached:          63032 kB
SwapCached:          0 kB
Active:         189776 kB
Inactive:        17136 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:       252444 kB
```

```
LowFree:          6428 kB
SwapTotal:      589816 kB          //交换分区约为内存的 2～2.5 倍
SwapFree:       589708 kB
Dirty:             36 kB
Writeback:          0 kB
AnonPages:      142076 kB
Mapped:          45352 kB
Slab:            19172 kB
PageTables:       5960 kB
NFS_Unstable:        0 kB
Bounce:              0 kB
CommitLimit:    716036 kB
Committed_AS:   526868 kB
VmallocTotal:   598008 kB
VmallocUsed:      4356 kB
VmallocChunk:   593420 kB
```

3. 查看系统模块信息

例如查看系统当前加载了哪些模块，可以使用下面的命令：

```
# cat   modules
bridge 53725 0 - Live 0xd0c6c000
netloop 10817 0 - Live 0xd0bd3000
netbk 78017 0 [permanent], Live 0xd0c46000
blktap 385125 2 [permanent], Live 0xd0bdc000
blkbk 21089 0 [permanent], Live 0xd0b9c000
autofs4 23749 2 - Live 0xd0b84000
hidp 23105 2 - Live 0xd0ace000
rfcomm 42457 0 - Live 0xd0bc7000
l2cap 29505 10 hidp,rfcomm, Live 0xd0b65000
bluetooth 53925 5 hidp,rfcomm,l2cap, Live 0xd0b8d000          //蓝牙设备支持
sunrpc 142973 1 - Live 0xd0ba3000
ip_conntrack_ftp 11697 0 - Live 0xd0b61000
ip_conntrack_netbios_ns 6977 0 - Live 0xd0abc000
ipt_REJECT 9537 1 - Live 0xd0b5d000
xt_state 6209 12 - Live 0xd0ac5000
ip_conntrack 53153 3 ip_conntrack_ftp,ip_conntrack_netbios_ns,xt_state, Live 0xd0b6e000
nfnetlink 10713 1 ip_conntrack, Live 0xd0aca000
iptable_filter 7105 1 - Live 0xd0ab9000                        //iptalbe 包过滤防火墙
ip_tables 17029 1 iptable_filter, Live 0xd0abf000
ip6t_REJECT 9409 1 - Live 0xd0ab5000
xt_tcpudp 7105 30 - Live 0xd0a9a000
ip6table_filter 6849 1 - Live 0xd0a45000                       //ip6talbe 防火墙
ip6_tables 18181 1 ip6table_filter, Live 0xd0aaf000
x_tables 17349 6 ipt_REJECT,xt_state,ip_tables,ip6t_REJECT,xt_tcpudp,ip6_tables, Live 0xd0aa9000
video 19269 0 - Live 0xd0aa3000
sbs 18533 0 - Live 0xd0a9d000
i2c_ec 9025 1 sbs, Live 0xd0a30000
button 10705 0 - Live 0xd0a24000
battery 13637 0 - Live 0xd0a91000
asus_acpi 19289 0 - Live 0xd0a8b000
```

```
ac 9157 0 - Live 0xd0a28000
ipv6 251137 15 ip6t_REJECT, Live 0xd0ad5000
lp 15849 0 - Live 0xd093b000                    //打印机 15849
sg 35933 0 - Live 0xd09ad000
floppy 54949 0 - Live 0xd0a36000                //软磁盘
snd_ens1371 28385 1 - Live 0xd09fe000
gameport 18889 1 snd_ens1371, Live 0xd09f8000
snd_rawmidi 26945 1 snd_ens1371, Live 0xd09f0000
snd_ac97_codec 87137 1 snd_ens1371, Live 0xd0a0d000     //ac97 声卡模块
snd_ac97_bus 6337 1 snd_ac97_codec, Live 0xd0994000
snd_seq_dummy 7877 0 - Live 0xd098b000
snd_seq_oss 32705 0 - Live 0xd09b8000
snd_seq_midi_event 11073 1 snd_seq_oss, Live 0xd0987000
snd_seq 49841 5 snd_seq_dummy,snd_seq_oss,snd_seq_midi_event, Live 0xd09e2000
snd_seq_device 11853 4 snd_rawmidi,snd_seq_dummy,snd_seq_oss,snd_seq, Live 0xd095d000
snd_pcm_oss 42849 0 - Live 0xd09d6000
snd_mixer_oss 19137 1 snd_pcm_oss, Live 0xd098e000
snd_pcm 72005 3 snd_ens1371,snd_ac97_codec,snd_pcm_oss, Live 0xd09c3000
snd_timer 25029 2 snd_seq,snd_pcm, Live 0xd09a5000
pcspkr 7105 0 - Live 0xd095a000
snd_ens1371,snd_rawmidi,snd_ac97_codec,snd_seq_oss,snd_seq,snd_seq_device,snd_pcm_oss,snd_mixer_o
ss,snd_pcm,snd_timer, Live 0xd0997000
i2c_piix4 12109 0 - Live 0xd0929000
soundcore 13217 1 snd, Live 0xd0955000
pcnet32 36805 0 - Live 0xd097d000
snd_page_alloc 13769 1 snd_pcm, Live 0xd087b000
i2c_core 23745 2 i2c_ec,i2c_piix4, Live 0xd0976000
mii 9409 1 pcnet32, Live 0xd0925000
parport_pc 29157 1 - Live 0xd096d000
parport 37641 2 lp,parport_pc, Live 0xd0962000
serial_core 23617 0 - Live 0xd0934000
ide_cd 40033 0 - Live 0xd094a000
cdrom 36705 1 ide_cd, Live 0xd0940000           //光盘驱动器
serio_raw 10693 0 - Live 0xd08e2000
dm_snapshot 20581 0 - Live 0xd092d000
dm_zero 6209 0 - Live 0xd0863000
dm_mirror 29840 0 - Live 0xd0826000
dm_mod 56537 8 dm_snapshot,dm_zero,dm_mirror, Live 0xd0906000
mptspi 20169 2 - Live 0xd0875000
mptscsih 26177 1 mptspi, Live 0xd085b000
mptbase 53089 2 mptspi,mptscsih, Live 0xd0917000
scsi_transport_spi 26177 1 mptspi, Live 0xd0853000
sd_mod 22977 3 - Live 0xd083e000
scsi_mod 130893 5 sg,mptspi,mptscsih,scsi_transport_spi,sd_mod, Live 0xd08c1000    //SCSI 设备
ext3 123081 2 - Live 0xd08e6000                                                     //ext3 设备
jbd 56553 1 ext3, Live 0xd0866000
ehci_hcd 33229 0 - Live 0xd0849000
ohci_hcd 23645 0 - Live 0xd0837000
uhci_hcd 25677 0 - Live 0xd082f000
```

4. 查看系统的版本信息

例如查看当前系统的版本，可以使用下面命令：

```
# cat  version
Linux version 2.6.18-8.el5xen (brewbuilder@ls20-bc2-14.build.redhat.com) (gcc version 4.1.1 20070105 (Red Hat 4.1.1-52)) #1 SMP Fri Jan 26 14:42:21 EST 2007
```

5. 查看系统分区信息

例如查看当前系统分区，可以使用下面命令：

```
# cat   partitions
major minor  #blocks  name

   8     0    8388608 sda          //SCSI 设备
   8     1     104391 sda1         // SCSI 设备分区 1
   8     2    8281507 sda2         // SCSI 设备分区 2
 253     0    7667712 dm-0
 253     1     589824 dm-1
```

8.1.2　系统监视器

用户可以通过【系统监视器】来查看系统资源（包括 CPU、内存、磁盘空间等）的使用情况。在面板中选择【系统】菜单，然后选择【管理】，在弹出的级联菜单中选择【系统监视器】，打开【系统监视器】对话框，选择【资源】选项卡，如图 8.1 所示。

图 8.1　查看系统资源情况

其中列出了 CPU 使用的历史情况，CPU 占用率，内存和网络的使用情况。单击【进程】选项卡，可以查看当前系统中进程的 PID 号、CPU 占用率、状态、优先级等信息，同时还可以看到系统前一分钟、五分钟、十五分钟的平均负载，如图 8.2 所示。

图 8.2 查看当前系统中进程

在【File system】选项卡中列出了当前已加载的文件系统。

8.1.3　磁盘使用分析器

用户可以使用【磁盘使用分析器】查看系统磁盘的使用情况。在面板中选择【应用程序】菜单，然后在弹出的菜单中选择【系统工具】，单击级联菜单中的【磁盘使用分析器】选项，打开如图 8.3 所示【磁盘使用分析器】窗口，其中显示了已经使用的磁盘空间大小。

图 8.3【磁盘使用分析器】窗口

在图 8.3 的菜单栏中选择【分析器】菜单，然后选择【扫描文件系统】选项，在目录树选项卡中会列出当前系统的目录结构和空间使用情况，如图 8.4 所示。

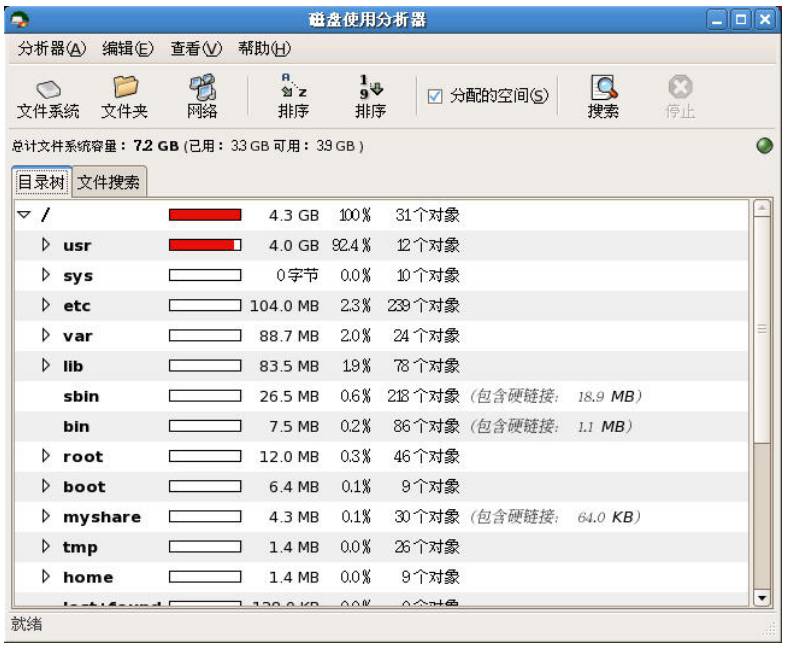

图 8.4 扫描文件系统

8.2 查看用户使用系统资源情况

8.2.1 w 命令

在多用户环境下，每个用户都可以登录到系统，执行不同的程序。利用 w 命令可以查看当前有哪些用户已经登录，以及正在进行什么操作。例如：

```
# w
15:19:50 up 2 days, 16:40,  3 users,  load average: 1.13, 0.34, 0.22
USER     TTY      FROM      LOGIN@   IDLE   JCPU   PCPU    WHAT
root     pts/1    :0.0      Tue23    0.00s  20.99s  4:13   gnome-terminal
root     pts/2    :0.0      Wed10    2days   0.52s  0.52s     bash
student   tty2     -        Wed11    1.00s   0.00s  0.04s     ping
```

其中显示系统当前时间为 15:19:50，即 w 命令被执行的时间；“up 2 days, 16:40”表示该系统已经运行了 2 天 16 个小时 40 分钟；“3 users”表示当前共有 3 位用户登录此系统；“1.13, 0.34, 0.22”表示系统在前 1 分钟，5 分钟，15 分钟的平均负载。其值越小表示系统负载越低，性能越佳。第二行共 8 个字段，用来显示用户正在进行的操作、占用的系统资源等情况，详细字段说明表如 8-1 所示。

表 8-1 w命令显示输出中各字段说明

字段	说明
USER	登录系统的用户名。
TTY	用户登录的终端。
FROM	显示用户从何处登录，如果是本地登录，此字段为“-”，如果是远程登录，则显示远程主机的主机名或IP地址，其中“:0.0”表示该用户是从X-Window以命令行模式登录。
LOGIN@	显示用户登录系统时的时间。
IDLE	显示用户空闲时间。
JCPU	显示与该终端相关的所有进程所消耗的CPU时间。

PCPU	表示CPU执行程序所消耗的时间。
WHAT	显示用户正在执行的程序的名称，如果正在执行命令行模式下的命令，则显示用户环境名称。

如果只查询指定用户的信息，可以在命令 w 后指定用户名，例如：

```
# w    root
15:29:50 up 2 days, 16:50,   2 users,   load average: 1.13, 0.32, 0.21
USER     TTY      FROM          LOGIN@     IDLE    JCPU     PCPU      WHAT
root     pts/1    :0.0           Tue23     0.00s   20.99s    4:13     gnome-terminal
root     pts/2    :0.0           Wed10     2days    0.52s   0.52s        bash
```

8.2.2　who 命令

使用命令 who 可以查看系统当前有哪些用户登录，例如：

```
# who
root     pts/1        2007-08-07 23:50 (:0.0)
root     pts/2        2007-08-08 10:31 (:0.0)
```

8.2.3　last 命令

使用命令 last 可以查看最近有哪些用户登录过系统，例如：

```
# last
root     pts/2        :0.0                 Wed Aug   8 10:31     still logged in
root     pts/1        :0.0                 Tue Aug   7 23:50     still logged in
root     :0                                Tue Aug   7 22:52     still logged in
reboot   system boot  2.6.18-8.el5xen     Tue Aug   7 22:43         (2+17:13)
...   ...
wtmp begins Thu Aug   2 07:46:13 2007
```

8.2.4　ac 命令

不带任何选项的 ac 命令可以查看系统总的连接时间，例如：

```
# ac
        total      240.03
```

其中默认的时间单位是小时。使用选项“-p”可以列出所有用户的连接时间，例如：

```
# ac   -p
        root                                  200.02
        student                                40.01
        total      240.03
```

可以使用选项“-d”，按时间对连接进行汇总，如下所示：

```
# ac   -d
Aug   2   total        16.23
Aug   3   total         0.70
Aug   4   total         9.36
Aug   5   total        30.90
Aug   7   total         1.28
Aug   8   total        61.47
Today     total       120.09
```

8.3　利用自动作业程序实现系统自维护

在 Red Hat Enterprise Linux 系统中可以使用“自动作业程序”来设置系统在某个时间执行特定的命令和进程。自动作业程序可以帮助系统管理员自动地执行数据备份、病毒扫描、检查邮箱，删除不必要的文件等工作。通过定期执行设定的任务，系统可以实现自动更新和维护。自动作业程序包括“Cron”和“Anacron”。

8.3.1　Cron 程序

Cron 是 Linux 中一个重要的 Daemon（守护进程），启动后该程序会常驻内存并定期启动设定的程序。

1. Cron 程序的安装与启动

检查系统中是否已安装 Cron 服务程序，可以使用下面的命令：

```
# rpm  -qa  vixie-cron
vixie-cron-4.1-66.1.el5    //已安装，且安装版本号为 4.1
```

启动 Cron 服务程序，可以使用下面的命令：

```
# /sbin/service  crond  start
Starting crond:                                          [  OK  ]
```

停止 Cron 服务程序，可以使用下面的命令：

```
# /sbin/service  crond  stop
Stopping crond:                                          [  OK  ]
```

如果 Cron 服务程序已经启动，则会显示如下错误信息：

```
# /sbin/service crond start
Starting crond: cannot start crond: crond is already runnin[FAILED]
```

2. 设置 Cron 任务

Cron 程序的任务设置可以通过编辑/etc/crontab 文件或者直接使用“crontab -e”命令。若使用的是“crontab -e”命令进行编辑，则在编写完毕后，系统默认的存储位置是/tmp。可以通过使用“crontab /etc/crontab”重新指定到/etc/crontab。下面是默认的 crontab 文件内容：

```
# vi  /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts      //下面部分是 Cron 定期执行的程序
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

其中共包括 9 行记录，第 1 行表示执行任务时使用的 Shell；第 2 行表示执行任务时系统的搜索路径；第 3 行表示将执行结果 mail 给 root；第 4 行指定主目录为根；第 5 行是注释；每 6 行表示每小时的第 1 分钟，以 root 身份执行/etc/cron.hourly 中的所有执行文件；第 7 行表示每天的 4 点 02 分以 root 身份执行/etc/cron.daily 中的所有执行文件；第 8 行表示每周日的的 4 点 22 分以 root 身份执行/etc/cron.weekly 中的所有执行文件；第 9 行表示每月 1 日的 4 点 42 分以 root 身份执行/etc/cron.monthly 中的所有执行文

件。

在/etc/crontab 文件中可以对定期启动的任务进行设定，格式为：

```
[分钟]  [小时]  [日期]  [月份]  [星期]  [用户]  [命令]
```

其中包括 7 个字段，各字段之间要以空格或 Tab 键隔开。每一行必须以回车结束，如果不加回车，Cron 程序将忽略该行，不执行该行设定的任务。其中各字段说明见表 8-2。

表 8-2 crontab文件各字段说明

字段	说明	示例
分钟	每小时的第几分钟执行	取值为0～59
小时	每天的第几小时执行	取值为0～23
日期	每月的第几天执行	取值为0～31
月分	每年的第几个月执行	取值为0～12或英文缩写，如：May、Feb、Nov等
星期	每周的第几天执行	取值为0～6或英文缩写，如：Sun、Mon、Tue等
用户	执行该命令的用户身份	root等
命令	定期执行的命令	如显示时间：date

在时间域中，可以使用“-”符号代表一段时间，例如在小时域中输入“6-12”，表示每小时的 6、7、8、9、10、11、12 分钟；可以使用“*”表示全部时间，例如在日期字段输入“*”，则表示每一个月的每一天都执行该命令；使用“,”表示特定的时间，例如在月份字段输入“3,5,12”，则表示每一年的 3 月、5 月和 12 月；使用“/”表示“每隔”，例如在分钟字段输入“*/5”，则表示每隔 5 分钟。

在修改了 crontab 文件之后不需要重新启动 Cron 服务程序。Cron 服务程序会自动根据 crontab 文件的内容刷新任务列表。

3. crontab 命令

crontab 命令格式如下：

```
crontab [ -u  用户名 ] 文件 1
crontab [ -u  用户名 ] [ -e | -l | -r ]
```

- -e：编辑用户的 crontab 文件。
- -l：列出用户在 crontab 中设定的任务。
- -r：删除用户在 crontab 中设定的任务。
- -i：删除用户设定的任务前进行提示。

例如使用“crontab　-e”添加新的任务，要求每隔 5 分钟将系统时间写入/myshare/cron_test 文件中，编辑命令行为：

```
*/5 * * * * date >> /myshare/cron_test
```

查看/myshare/cron_test 文件，可以看到时间已经自动写入该文件：

```
# cat   /myshare/cron_test
Wed Aug   8 07:08:02 CST 2007
Wed Aug   8 07:09:01 CST 2007
```

例如在每月的 3 日 23:30 自动删除/var/log/httpd 目录下的所有文件，编辑命令行为：

```
30 23 3 * *   root   rm   -f   /var/log/httpd/*
```

例如每隔 5 分钟查询一次系统当前运行的进程，并将查询结果保存到/myshare/cron_test 文件中，编辑命令行为：

```
*/5 * * * * *   ps   -aux > /myshare/cron_test
```

例如每周一的 1：00、3：00、8：00 各查询一次根目录结构，并将查询结果保存到/myshare/cron_test 文件中，编辑命令行为：

```
0 1,3,8 * * Mon   ls   –al   /   >   /myshare/cron_test
```

不带“-u”选项使用 crontab 命令，系统默认为当前用户创建任务，如果为其他用户设定任务只需

在“-u”后指定用户。例如编辑用户 teacher1 的任务，可以使用命令：

```
#crontab  -e  -u  teacher1
```

使用“-l”选项可以列出所有的任务列表。例如列出 root 用户的所有任务列表，命令如下：

```
# crontab  -l  -u root
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
*/1 * * * * date >> /myshare/cron_test
42 4 1 * * root run-parts /etc/cron.monthl
```

4. cron.allow 与 cron.deny

为了增强 Cron 服务程序的安全性，系统使用/etc/cron.allow 和/etc/cron.deny 文件对 crontab 文件的存取进行管理。如果/etc/cron.allow 文件存在，则只有该文件列出的用户可以使用 Cron。如果/etc/cron.deny 文件存在，则在该文件中列出的用户不可以使用 Cron。对于根用户，无论是否包含在这两个文件中，都有使用 Cron 服务的权力。

5. 利用 Cron 程序清除垃圾文件

Cron 服务程序被经常用来帮助系统管理员定期清除垃圾文件。例如 Linux 中的 core 文件经常占用大量磁盘空间，可以编辑 crontab 文件，实现每周一早上 2:00 自动删除所有一周以来没有访问过的 core 文件：

```
0 2 * * Mon  find  /  -name  core  -atime  +7  -exec  rm  -f  {}  \;          //  “}”与”\”之间要有空格
```

8.3.2　Anacron 程序

Anacron 程序与 Cron 程序类似，也属于任务调度工具。Anacron 设定的任务如果在指定时间没有成功执行，Anacron 会间隔一段时间后再次执行该任务，而 Cron 设定的任务如果在指定时间没能完成，则调度工作就会失败。因而 Anacron 设定的任务的执行机率要比 Cron 高。

1. Anacron 程序的安装与启动

使用 Anacron 前应检查系统中是否已安装 Anacron 服务程序，可以使用下面的命令：

```
# rpm  -qa  anacron
anacron-2.3-45.el5          //已安装
```

启动 Anacron 服务程序，可以使用下面的命令：

```
# /sbin/service anacron  start
Starting anacron:                                          [  OK  ]
```

若要停止 Anacron 服务程序，可以使用下面的命令：

```
# /sbin/service anacron  stop
Stopping anacron:                                          [  OK  ]
```

若要重新启动 Anacron 服务程序，可以执行下面命令：

```
# /sbin/service anacron  restart
Stopping anacron:                                          [  OK  ]
Starting anacron:                                          [  OK  ]
```

2. 设置 acnacron 任务

设置 Anacron 服务程序可以通过编辑/etc/anacrontab 文件实现。文件中除了注释和环境变量外，每一行表示一个任务，基本要求与 Cron 相同，设定格式如下：

```
[时间间隔]  [等待时间]  [任务标识]  [命令]
```

其中：

- [时间间隔]：指执行任务的时间间隔，以日为单位进行计算。
- [等待时间]：指时间间隔到期后，由于主机没有正常开机等原因导致任务没有顺利执行，则等待一段时间后尝试再次执行，以秒为单位进行计算。
- [任务标识]：记录此任务相关的说明。
- [命令]：设定执行的程序。

下面是默认的/etc/anacrontab 文件内容：

```
#cat anacrontab              //以下是 anacrontab 文件内容
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
1        65        cron.daily                run-parts /etc/cron.daily
7        70        cron.weekly               run-parts /etc/cron.weekly
30       75        cron.monthly              run-parts /etc/cron.monthly
```

其中共 8 行，其中第 1、2 行是注释，表明该文件是 Anacron 的配置文件，详细使用说明可以查看 anacron（8）或 anacrontab（5）；第 3 行表示默认的 Shell；第 4 行是执行该任务时，默认的搜索路径；第 5 行表示执行结果以电子邮件方式传送给根用户；第 6 行表示每隔 1 天执行/etc/cron.daily 目录下的程序，如果未能按时执行，则间隔 65 分钟后继续尝试；第 7 行表示每隔 7 天执行/etc/cron.weekly 目录下的程序，如果未能按时执行，则间隔 70 分钟后继续尝试；第 8 行表示每隔 30 天执行/etc/cron.monthly 目录下的程序，如果未能按时执行，则间隔 75 分钟后继续尝试。

8.3.3　at 程序

at 程序也是一种任务管理工具，允许用户将一个或多个命令放到队列中，按时执行。与 Cron 不同的是，at 设置的任务只在某个时刻执行，并且只执行一次。如果要使用 at 调度任务，必须首先启动 atd 守护进程，命令如下：

```
#/usr/sbin/atd
```

启动 atd 守护进程后，就可使用 at 命令设置任务。例如设置 21 小时 55 分将根目录下的所有文件的列表保存到/myshare/at_test 文件里，命令行为：

```
# at   21:55
at> ls   -al   /   >   /myshare/at_test
at> <EOT>                                  //Ctrl+d 退出
job 2 at 2007-08-09 21:55
```

在键入“at 21:55”并按回车后，at 命令会显示一个专用的提示符“at>”，输入需要执行的命令之后按 Ctrl+d 退出 at，at 会提示任务执行的时间及编号。用户在“at>”提示符下可以输入多个命令，直到用户按下 Ctrl+d 组合键。

时间的设定可以采用多种格式，例如，8:50 p.m May07、11:00 pm、+10 days、tomorrow、now、noon、midnight 等。例如设置任务，要求 5 分钟后向/myshare/at_test 文件中存入单词“hello”，命令行为：

```
# at  now  +5  min
at> echo  "hello"  >>  /myshare/at_text
at> <EOT>
job 4 at 2007-08-09 09:59
# date                    //系统当前时间为 09:59
2007 年 08 月 09 日 星期四 09:59:17 CST
# cat /myshare/at_text
hello              //任务已执行完毕
```

使用 atq 命令可以查询当前已经设置的任务，命令行如下：

```
# atq
1        2007-08-09 21:40 a root
2        2007-08-09 21:55 a root
3        2007-08-09 22:57 a root
```

如果需要删除某个任务，可以使用 atrm 命令，该命令格式为：

```
atrm  N
```

其中 N 是任务编号，可以通过 atq 命令查询。例如删除当前任务队列中的第 3 个任务，命令行为：

```
# atq
1        2007-08-09 21:40 a root
2        2007-08-09 21:55 a root
3        2007-08-09 22:57 a root
# atrm 3
# atq
1        2007-08-09 21:40 a root
2        2007-08-09 21:55 a root
```

所有已被执行的任务都被存储在/var/spool/at 目录中，可以从该目录中查看设置的任务，如下所示：

```
# cd  /var/spool/at
# ls
a00001012dc8f4  a00002012dc903  spool
# cat a00001012dc8f4
#!/bin/sh
# atrun uid=0 gid=0
# mail       root 0
...  ...      //环境变量
umask 22
...  ...
du  -a  >  /myshare/at_text          //用户设置的任务
```

不是所有用户都有使用 at 程序的权限。at 的权限设置是通过/etc/at.allow 和/etc/at.deny 两个文件来完成的。如果 at.allow 文件存在，系统会只允许在该文件中列出的用户使用 at 程序，其他用户无权使用；如果只需要拒绝很少一部分用户，而允许其他用户，可以使用 at.deny 文件，所有在该文件中被列出的用户将无权使用 at；如果 at.allow 和 at.deny 文件都不存在，则除了根用户，其他用户均无权使用 at。

注意：at.allow 和 at.deny 文件中，每行只能输入一个用户名。

8.3.4　batch 命令

batch 命令实际上仍然使用的是 atd 守护进程，与 at 命令格式、用法也完全相同。不同之处在于由 batch 设置的任务会在系统平均负载低于一个特定值（默认为 0.8）时才会运行，可以有效防止调度的任务占用过多的处理器时间。例如利用命令 du 对磁盘空间的使用情况进行统计通常是一项极耗费系统资

源的工作，可以使用 batch 命令添加一个新任务，使该工作在平均负载比较低时才被执行，并将结果保存到/tmp/diskspace 文件中，命令行如下：

```
#batch
at> du  -h  /  > /tmp/diskspace
at><Ctrl+d>
job 6 at 2007-08-09 19:12
```

8.4　改变进程优先级

在 Linux 系统中，进程有运行、就绪和阻塞三种状态，进程会在三种状态间切换。每个正被执行的进行都会被赋予一定的使用 CPU 的优先级。系统会按着优先级对进程进行调度。等级越高的进程在执行时就会有更多地获得使用 CPU 的机会，其大部分时间都会处于运行态，总的执行时间就会缩短。与之相反，等级较低的进程就需要较长时间等待 CPU，因此总的执行时间就会越长。

8.4.1　nice 命令

在 Red Hat Enterprise Linux 中提供了 nice 和 renice 命令，可以根据用户的需要设置进程执行的优先级。

nice 的命令格式为：

```
nice  -N  命令
```

其中 N 为-20～19 之间的整数，表示进程执行时的优先等级。-20 表示最高等级，19 代表最低等级。其中-20～-1 之间的等级只有系统管理员可以设置。如果没有使用 nice 命令对进程执行时的优先级进行设定，系统默认等级为 0；如果使用了 nice 命令，但没有指定等级，则系统默认为 10。表 8-3 是 nice 命令使用范例及说明。

表 8-3 nice命令使用范例及说明

命令	优先级说明
vi test1 &	没有使用nice命令，因而系统默认赋予其优先级为0。
nice vi test2 &	使用nice命令，但没有指定优先级，系统默认赋予其优先级为10。
nice -25 vi test3 &	由于用nice命令指定的优先级25，超出系统最大值，系统会以最低等级19运行。
nice --25 vi test4 &	由于用nice命令指定的优先级-25，超出系统最小值，系统会以最高等级-20运行。
nice -15 vi test5 &	使用nice命令，并指定优先级为15，所以该进程的等级为15。
nice --15 vi test6 &	使用nice命令，并指定优先级为-15，所以该进程的等级为-15。

可以通过查看进程的优先级来验证上述结论：

```
# ps  -l  |  grep  vi
0 T     0  6087  3796  0  77   0 -   1239 finish pts/1    00:00:00 vi
0 T     0  6088  3796  0  87  10 -   1239 finish pts/1    00:00:00 vi
0 T     0  6124  3796  0  97  19 -   1239 finish pts/1    00:00:00 vi
4 T     0  6125  3796  0  60 -20 -   1240 finish pts/1    00:00:00 vi
0 T     0  6128  3796  0  90  15 -   1240 finish pts/1    00:00:00 vi
4 T     0  6129  3796  0  62 -15 -   1239 finish pts/1    00:00:00 vi
```

8.4.2　renice 命令

nice 命令用于设置优先级并执行相应的程序。如果一个程序已经开始执行，可以使用 renice 命令对优先级进行调整。renice 命令格式为：

```
renice　N　[ [进程号] [-u 用户名] [-g 用户组] ]
```

其中 N 为-20～19 之间的整数，表示进程的优先等级；[进程号]指进程的 PID 值；[-u]或[-g]用于改变用户或用户组中，所有进程执行的优先级。

注意：renice 使用进程号来对进程进行指定，而 nice 命令使用程序名。renice 命令的参数 N 前不需要加上“-”符号，而 nice 命令的参数 N 前需要加上“-”符号。

例如修改 vi 进程的优先级，命令行如下：

```
# ps　-l　| grep　vi
0 T　　0　6087　3796　0　77　　0 -　1239 finish pts/1　　00:00:00 vi
# renice　-10　6087
6087: old priority 0, new priority -10
```

如果重新修改的等级超出最小值，则该进程的优先级取最高等级-20：

```
# renice　-25　6087
6087: old priority -10, new priority -20
```

如果重新修改的等级超出最大值，则该进程的优先级取最低等级 19：

```
# renice　25　6087
6087: old priority -20, new priority 19
```

例如修改用户 teacher1 和 teacher2 所执行进程的优先级为-10，命令行如下

```
# renice　-10　-u　teacher1　teacher2
500: old priority -10, new priority -10
505: old priority　0, new priority -10
```

如果修改 teacher 用户组中所有用户执行的进程的优先级为-5，命令行如下：

```
# renice -5 -g teacher
0: old priority -10, new priority -5
```

8.4.3　使用系统监视器更改优先级

在 Red Hat Enterprise Linux 5 中可以使用【系统监视器】来更改进程的优先级。

（1）在【系统监视器】中选择需要改变优先级的进程，右键单击并在快捷菜单中选择【更改优先级】选项，如图 8.5 所示。

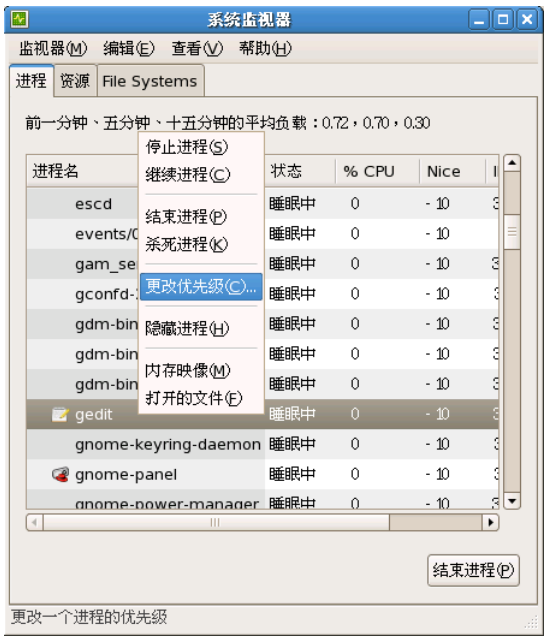

图 8.5 在快捷菜单中选择【更改优先级】选项

（2）在弹出的【改变优先级】菜单中，拉动水平滑杆可以设置新的优先等级，如图 8.6 所示。

图 8.6【改变优先级】对话框

8.5　Linux 系统日志

日志是实现 Linux 系统安全的重要手段，利用日志可以审计和监测系统出现的错误，侦察和追踪入侵，协助系统的恢复和排障。Red Hat Enterprise Linux 具有自动记录日志功能，通常使用 syslog 和 klog 来完成。syslog 记录常规系统日志，而 klog 针对内核活动进行记录。

日志会按类别记录在日志文件中，一般位于/var/log 目录下。其中/var/log/dmesg 文件记录了与启动 Linux 相关的基本引导信息，/var/log/messages 文件记录了系统引导之后的附加信息，/var/log/boot.log 文件记录了与启动和停止守护进程相关的信息，/var/log/wtmp 文件帮助监视系统登录情况。根用户可以使用文体编辑器（例如 gedit 或 VI）查看这些日志文件，也可以使用 Red Hat Enterprise Linux 5 的【系统日志查看器】间接查看日志文件。

8.5.1　启动 syslog 日志进程

几乎所有的 UNIX 系统都采用 syslog 进行系统日志。syslog 有两个重要的文件：/etc/syslogd（守护

进程）和/etc/syslog.conf 配置文件。启动 syslog 日志守护进程使用命令如下：

```
#/sbin/syslogd
```

如果希望守护进程可以接收来自网络的 syslog 信息，可以使用“-r”选项，命令行为：

```
#/sbin/syslogd  -r
```

通常在系统启动时，系统会自动启动 syslog 守护进程，重复运行“/etc/syslogd”命令，系统会提示已运行：

```
# syslogd
syslogd: Already running.
```

在重新修改配置文件后需要重启 syslog 进程才能使新的配置生效，命令行如下：

```
#killall  -HUP  syslogd
```

其中“-HUP”使 syslog 关闭所有日志文件，重读/etc/syslog.conf 配置文件后重新开始记录日志。

8.5.2　系统日志配置文件 syslog.conf

系统日志的主配置文件为 syslog.conf，存储于/etc 目录下。

1. syslog.conf 的语法格式

syslog 的执行内容由/etc/syslog.conf 文件设定，syslog.conf 的语法格式如下：

```
[消息设备.消息级别] [动作]
```

其中[消息来源.消息级别]和[动作]之间必须用 Tab 分隔，消息来源指发出消息的设备或程序，表 8-4 列出了常用的消息来源。

表 8-4 syslog常用消息来源

消息来源	说明
kern	内核
uucp	uucp程序
user	用户程序
news	Usenet系统消息
mail	邮件系统
daemon	守护进程
auth	与安全认证及权限修改相关的命令
syslog	syslog自身产生的消息
cron	cron程序产生的消息
mark	时间戳
authpriv	私有的授权信息
lpr	打印机
local 0-7	本地消息

消息级别指消息的紧急程度。如果在一行上出现多个[消息设备.消息级别]，各项之间需用分号分隔。例如，“kern.emerg”表示来自内核的紧急信息，“mail.*；daemon.noctice”表示所有与邮件相关的信息和来自守护进程的警告信息。常用的消息级别如表 8-5 所示，其中紧急程序由上到下逐级递减。

表 8-5 syslog常用消息级别

消息级别	说明
emerge	最高的紧急等级，指极度恐慌，与panic同意
alert	紧急状态
crit	临界状态
err	出现错误
warning	警告

notice	出现了不正常现象，可能需要检查
info	一般性消息
debug	调试信息

其中紧急程序遵循向上匹配原则。例如一个“err”选项表示所有大于等于“err”等级的消息都将被处理，即所处理的消息包括“err”、“crit”、“alert”、“emerge”；而如果被标志为“debug”，则所有消息都要被处理。如果只希望匹配某个确定的紧急程序，而不是向上匹配，需要使用等号进行设定。例如“kern.=alert”表示只对内核产生的紧急信息（alert）进行处理。

syslog.conf 文件的配置行也支持通配符“*”和“none”。其中“*”表示匹配全部，“none”表示忽略全部，例如“daemon.*”表示守护进程产生的所有信息，而“kern.none”将忽略内核的所有信息。

[动作]用于设定 syslog 如何处理对应的信息。可以设定将信息写入文件或显示在终端上，或直接发送给指定用户，也可以发送给指定的另一台主机。syslog 可用动作如表 8-6 所示。

表 8-6 syslog可用动作及说明

动作	说明
@主机名	转发给另一台主机上的syslog程序。
@IP地址	转发给另一个IP地址的主机。
*	转发到所有用户的终端上。
/dev/console	转发到本地主机的终端上。
| 程序	通过命名管道转发给程序。
文件名	将信息写入指定的文件（文件必须使用绝对路径）。
用户列表	将信息发给用户列表中的所有用户，用户名之间用逗号分隔。

2. syslog.conf 文件的默认设置

/etc/syslog.conf 文件的默认配置如下所示：

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console   //内核的所有信息发送到本地主机终端上
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!            // 不对私人认证信息进行日志
*.info;mail.none;news.none;authpriv.none;cron.none      /var/log/messages
# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure
# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog
# Log cron stuff
cron.*                                                  /var/log/cron
# Everybody gets emergency messages
*.emerg                                 *     //所有的 emerge 信息发送给所有登录的用户
# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler
# Save boot messages also to boot.log
local7.*                                /var/log/boot.log   //将启动信息同时记录到 boot.log 文件中
#
# INN
#
news.=crit          /var/log/news/news.crit   //将 Usernet 发出的 crit 信息记录到/var/log/news/news.crit 文件中
news.=err           /var/log/news/news.err    //将 Usernet 发出的 err 信息记录到/var/log/news/news.err
news.notice         /var/log/news/news.notice //将 Usernet 发出的 notice、warning、err、crit、alert、emerge 信息
                                              //记录到/var/log/news/news.notice
```

3. syslog.conf 文件的配置实例

例如将一般性的消息保存在/var/log/messages 文件中，但不包括邮件、新闻组、本地安全认证、守护进程、cron 程序所产生的信息，配置行如下：

```
*.info;mail.none;news.none;authrpiv.none;daemon.none;cron.none        /var/log/messages
```

例如发生内核恐慌（emerge），则将消息发送给所有登录的用户，配置行如下：

```
kern.emerg        *
```

例如对所有与电子邮件相关的信息保存到/var/log/maillog 文件中，配置行如下：

```
mail.*       /var/log/maillog
```

例如将与打印机相关的信息发送到 student.bit.edu.cn 主机上记录下来，配置行如下：

```
lpr.*       @student.bit.edu.cn
```

8.5.3　测试 syslog.conf

使用 logger 程序可以模拟各类消息，从而可以对 syslog.conf 文件的配置进行测试。用户按着配置文件 syslog.conf 中的设置，设定 logger 发出指定类型的消息，从而可以检测配置是否正确。logger 命令格式如下：

```
logger   -p   [消息] [消息内容]
```

例如测试 syslog.conf 中的“*.emerg　　*”，可以执行命令：

```
# logger   -p   kern.emerg   "Just test"
Message from syslogd@localhost at Fri Aug 10 13:34:46 2007 ...
localhost root: Just test
```

例如测试 syslog.conf 中的“mail.*　　/var/log/maillog”，可以执行命令：

```
# logger   -p   mail.info   "Just mail test"
# grep   test /var/log/maillog
Aug 10 13:38:35 localhost root: Just mail test
```

8.5.4　清空运行日志

随着系统运行时间越来越长，日志文件也会变得越来越大。如果创建日志文件的服务正在运行，必须停止服务，才能正常删除日志，否则会产生不可预测的后果。但可以利用 echo 命令清空日志，而不必停止日志服务。命令格式为：

```
#echo   >   日志文件
```

例如要清空保存在/usr/local/apache/logs/error_log 中的 Apache 服务器日志，可以运行：

```
#echo   >   /user/local/apache/logs/error_log
```

8.5.5　系统日志的图形化管理

（1）在面板的【系统】菜单中选择【管理】，在弹出的级联菜单中单击【系统日志】选项，打开【系统日志查看器】窗口（如果非根用户，则在打开【系统日志查看器】窗口之前系统会弹出验证对话框，输入正确的 root 口令即可），如图 8.7 所示。

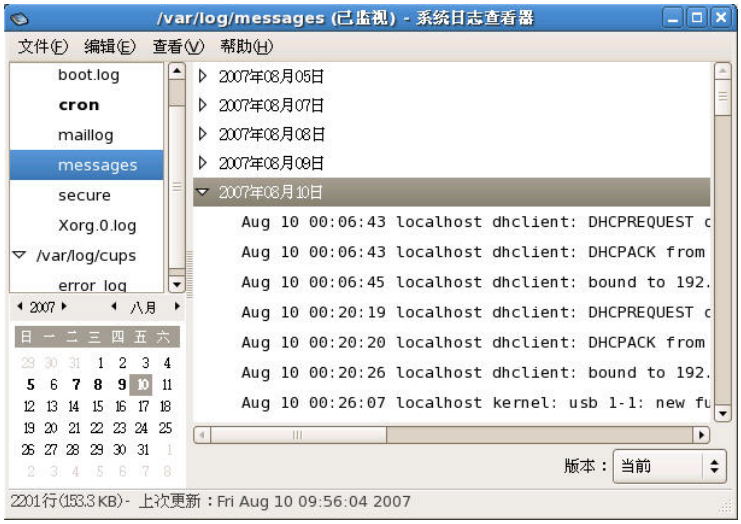

图 8.7【系统日志查看器】窗口

（2）在【系统日志查看器】窗口中可以查看分类日志，还可以利用【查看】菜单中的【过滤器】进行过滤显示。

8.5.6　使用日志进行故障诊断

1. 使用/var/log/dmesg 日志文件

在日志文件/var/log/dmesg 中记录了系统启动时内核是如何对硬件进行配置的：首先从 BIOS 开始，然后依次查找 CPU、硬盘驱动器、PCI 设备、通信端口，接下来启动分区上的文件系统，最后配置键盘和鼠标等其他设备。下面是 dmesg 日志文件的一部分：

```
Linux version 2.6.18-8.el5xen (brewbuilder@ls20-bc2-14.build.redhat.com) (gcc version 4.1.1 20070105 (Red Hat 4.1.1-52)) #1 SMP Fri Jan 26 14:42:21 EST 2007
BIOS-provided physical RAM map:
 Xen: 0000000000000000 - 000000000ffc7000 (usable)
0MB HIGHMEM available.
255MB LOWMEM available.
Using x86 segment limits to approximate NX protection
...  ...
```

其中可以看到系统中的内存为 255M，如果该计算机实际上安装了大于 255M 的扩展内存，则从该日志可以判断新加入的内存没能被 Linux 系统识别。

在该日志文件的尾部如果可以看到下面的记录：

```
EXT3 FS on dm-0, internal journal
kjournald starting.  Commit interval 5 seconds
EXT3 FS on sda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
```

该信息表明，Linux 系统已经将一个 ext3 文件系统挂载到了一个分区上。

2. 使用/var/log/cron 日志文件

/var/log/cron 日志文件中记录了 cron 作业的执行时间和日期，如下所示：

```
Aug   8 07:11:11 localhost crontab[6923]: (root) LIST (root)
Aug   8 07:12:01 localhost crond[6765]: (root) RELOAD (cron/root)
```

```
Aug   8 07:12:01 localhost crond[6929]: (root) CMD (date >> /myshare/cron_test)
Aug   8 07:13:01 localhost crond[6935]: (root) CMD (date >> /myshare/cron_test)
...   ...
```

黑客在进行入侵活动时经常会用到 cron 程序，例如通过 cron 程序定期打开后门以方便入侵系统。如果在该日志文件中发现有未经授权的程序在定期执行，应加以格外注意并采取必要的防范手段。

在/var/log 目录中还包括许多其他的日志文件，表 8-7 列出了部分日志文件和功能说明。

表 8-7 /var/log日志文件及功能说明

日志文件	功能说明
cups	与打印服务相关的日志文件目录
gdm	存放GNOME启动日志文件的目录
httpd	存放Web服务器日志文件的目录
news	存放与InterNetNews服务相关的日志文件目录
squid	存放Squid代理服务日志文件的目录
secure	与安全连接相关的日志文件目录
scrollkeeper.log	用于文档，尤其是GUI中的文档
boot.log	记录了与启动和停止守护进程相关的信息
ppp	存放与ppp相关的日志文件目录
rpmpkgs	当前已安装的RPM程序包文件
lastlog	最近登录系统的用户
samba	与samba服务相关的日志文件目录
dmesg	记录了与启动Linux相关的引导信息
wtmp	记录用户登录系统情况