# [linux 中/proc 详解](https://www.cnblogs.com/aofo/p/6151266.html)

proc 文件系统  
   
在Linux中有额外的机制可以为内核和内核模块将信息发送给进程-- /proc 文件系统。最初设计的目的是允许更方便的对进程信息进行访问（因此得名），现在它被每一个有有趣的东西报告的内核使用，例如/proc/modules 有模块的列表/proc/meminfo 有内存使用的统计表。  
   
　　使用proc 文件系统的方法和使用设备驱动程序非常相似--创建一个/proc 文件需要的所有信息的结构，包括任何处理函数的指针（在我们的例子中只有一个，当某人试图从/proc 文件读时调用的那一个）。然后，init\_module 在内核中登记该结构而cleanup\_module 注销它。  
   
　　我们使用proc\_register\_dynamic(这是在2.0 版中的情况，在2.2 版中如果我们将节点设置为0系统将自动为我们做到) 的原因是我们不想预先决定我们的文件的节点数字，而是为防止冲突而由内核决定它。通常的文件系统存在于磁盘上而不是内存中（/proc 在内存中），在这中情况下，节点数是是指向文件的索引节点所在的磁盘位置的指针。节点包含文件的信息（例如文件的存取权限）和指向磁盘位置或文件数据可以被找到的几个位置的指针。  
   
　　因为当文件被打开或关闭的时候不能得到调用，所以在这个模块中没有地方放置MOD\_INC\_USE\_COUNT 和MOD\_DEC\_USE\_COUNT，并且，如果文件被打开随后模块被移除，我们没有办法避免后果。在下一章我们会看到一个艰难的但更灵活的可以处理/proc文件的实现方式，它也可以让我们防止那个问题。  
   
http://www.2cto.com/os/201202/119552.html  
   
Linux 内核提供了一种通过/proc 文件系统，在运行时访问内核内部数据结构、改变内核设置的机制。尽管在各种硬件平台上的Linux 系统的/proc 文件系统的基本概念都是相同的，但本文只讨论基于intel x86 架构的Linux /proc 文件系统。  
   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   
/proc --- 一个虚拟文件系统  
   
/proc 文件系统是一种内核和内核模块用来向进程(process) 发送信息的机制(所以叫做/proc)。这个伪文件系统让你可以和内核内部数据结构进行交互，获取 有关进程的有用信息，在运行中(on the fly) 改变设置(通过改变内核参数)。 与其他文件系统不同，/proc 存在于内存之中而不是硬盘上。如果你察看文件/proc/mounts (和mount 命令一样列出所有已经加载的文件系统)，你会看到其中 一行是这样的：  
   
grep proc /proc/mounts  
/proc /proc proc rw 0 0  
   
/proc 由内核控制，没有承载/proc 的设备。因为/proc 主要存放由内核控制的状态信息，所以大部分这些信息的逻辑位置位于内核控制的内存。对/proc 进行一次'ls -l' 可以看到大部分文件都是0 字节大的；不过察看这些文件的时候，确实可以看到一些信息。这怎么可能？这是因为/proc 文件系统和其他常规的文件系统一样把自己注册到虚拟文件系统层(VFS) 了。然而，直到当VFS 调用它，请求文件、目录的i-node 的时候，/proc 文件系统才根据内核中的信息建立相应的文件和目录。  
   
加载proc 文件系统  
   
如果系统中还没有加载proc 文件系统，可以通过如下命令加载proc 文件系统：  
   
mount -t proc proc /proc  
上述命令将成功加载你的proc 文件系统。更多细节请阅读mount 命令的man page。  
   
察看/proc 的文件  
   
/proc 的文件可以用于访问有关内核的状态、计算机的属性、正在运行的进程的状态等信息。大部分/proc 中的文件和目录提供系统物理环境最新的信息。尽管/proc 中的文件是虚拟的，但它们仍可以使用任何文件编辑器或像'more', 'less'或'cat'这样的程序来查看。当编辑程序试图打开一个虚拟文件时，这个文件就通过内核中的信息被凭空地(on the fly) 创建了。这是一些我从我的系统中得到的一些有趣结果：  
   
$ ls -l /proc/cpuinfo  
-r--r--r-- 1 root root 0 Dec 25 11:01 /proc/cpuinfo  
   
$ file /proc/cpuinfo  
/proc/cpuinfo: empty  
   
$ cat /proc/cpuinfo  
   
processor : 0  
vendor\_id : GenuineIntel  
cpu family : 6  
model : 8  
model name : Pentium III (Coppermine)  
stepping : 6  
cpu MHz : 1000.119  
cache size : 256 KB  
fdiv\_bug : no  
hlt\_bug : no  
sep\_bug : no  
f00f\_bug : no  
coma\_bug : no  
fpu : yes  
fpu\_exception : yes  
cpuid level : 2  
wp : yes  
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca  
cmov pat pse36 mmx fxsr xmm  
bogomips : 1998.85  
   
processor : 3  
vendor\_id : GenuineIntel  
cpu family : 6  
model : 8  
model name : Pentium III (Coppermine)  
stepping : 6  
cpu MHz : 1000.119  
cache size : 256 KB  
fdiv\_bug : no  
hlt\_bug : no  
sep\_bug : no  
f00f\_bug : no  
coma\_bug : no  
fpu : yes  
fpu\_exception : yes  
cpuid level : 2  
wp : yes  
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca  
cmov pat pse36 mmx fxsr xmm  
bogomips : 1992.29  
   
这是一个从双CPU 的系统中得到的结果，上述大部分的信息十分清楚地给出了这个系统的有用的硬件信息。有些/proc 的文件是经过编码的，不同的工具可以被用来解释这些编码过的信息并输出成可读的形式。这样的工具包括：'top', 'ps', 'apm' 等。  
   
得到有用的系统/内核信息  
   
proc 文件系统可以被用于收集有用的关于系统和运行中的内核的信息。下面是一些重要的文件：  
   
\* /proc/cpuinfo - CPU 的信息(型号, 家族, 缓存大小等)  
\* /proc/meminfo - 物理内存、交换空间等的信息  
\* /proc/mounts - 已加载的文件系统的列表  
\* /proc/devices - 可用设备的列表  
\* /proc/filesystems - 被支持的文件系统  
\* /proc/modules - 已加载的模块  
\* /proc/version - 内核版本  
\* /proc/cmdline - 系统启动时输入的内核命令行参数  
   
proc 中的文件远不止上面列出的这么多。想要进一步了解的读者可以对/proc 的每一个文件都'more'一下或读参考文献[1]获取更多的有关/proc 目录中的文件的信息。我建议使用'more'而不是'cat'，除非你知道这个文件很小，因为有些文件(比如kcore) 可能会非常长。  
   
有关运行中的进程的信息  
   
/proc 文件系统可以用于获取运行中的进程的信息。在/proc 中有一些编号的子目录。每个编号的目录对应一个进程id (PID)。这样，每一个运行中的进程/proc 中都有一个用它的PID 命名的目录。这些子目录中包含可以提供有关进程的状态和环境的重要细节信息的文件。让我们试着查找一个运行中的进程。  
   
$ ps -aef | grep mozilla  
root 32558 32425 8 22:53 pts/1 00:01:23 /usr/bin/mozilla  
   
上述命令显示有一个正在运行的mozilla 进程的PID 是32558。相对应的，/proc 中应该有一个名叫32558 的目录  
   
$ ls -l /proc/32558  
total 0  
-r--r--r-- 1 root root 0 Dec 25 22:59 cmdline  
-r--r--r-- 1 root root 0 Dec 25 22:59 cpu  
lrwxrwxrwx 1 root root 0 Dec 25 22:59 cwd -> /proc/  
-r-------- 1 root root 0 Dec 25 22:59 environ  
lrwxrwxrwx 1 root root 0 Dec 25 22:59 exe -> /usr/bin/mozilla\*  
dr-x------ 2 root root 0 Dec 25 22:59 fd/  
-r--r--r-- 1 root root 0 Dec 25 22:59 maps  
-rw------- 1 root root 0 Dec 25 22:59 mem  
-r--r--r-- 1 root root 0 Dec 25 22:59 mounts  
lrwxrwxrwx 1 root root 0 Dec 25 22:59 root -> //  
-r--r--r-- 1 root root 0 Dec 25 22:59 stat  
-r--r--r-- 1 root root 0 Dec 25 22:59 statm  
-r--r--r-- 1 root root 0 Dec 25 22:59 status  
   
文件"cmdline" 包含启动进程时调用的命令行。"envir" 进程的环境变两。"status" 是进程的状态信息，包括启动进程的用户的用户ID (UID) 和组ID(GID) ，父进程ID (PPID)，还有进程当前的状态，比如"Sleelping"和"Running"。每个进程的目录都有几个符号链接，"cwd"是指向进程当前工作目录的符号链接，"exe"指向运行的进程的可执行程序，"root"指向被这个进程看作是根目录的目录(通常是"/")。目录"fd"包含指向进程使用的文件描述符的链接。"cpu"仅在运行SMP 内核时出现，里面是按CPU 划分的进程时间。  
   
/proc/self 是一个有趣的子目录，它使得程序可以方便地使用/proc 查找本进程地信息。/proc/self 是一个链接到/proc 中访问/proc 的进程所对应的PID 的目录的符号链接。  
   
通过/proc 与内核交互  
   
上面讨论的大部分/proc 的文件是只读的。而实际上/proc 文件系统通过/proc 中可读写的文件提供了对内核的交互机制。写这些文件可以改变内核的状态，因而要慎重改动这些文件。/proc/sys 目录存放所有可读写的文件的目录，可以被用于改变内核行为。  
   
/proc/sys/kernel - 这个目录包含反通用内核行为的信息。/proc/sys/kernel/{domainname, hostname} 存放着机器/网络的域名和主机名。这些文件可以用于修改这些名字。  
   
$ hostname  
machinename.domainname.com  
   
$ cat /proc/sys/kernel/domainname  
domainname.com  
   
$ cat /proc/sys/kernel/hostname  
machinename  
   
$ echo "new-machinename" > /proc/sys/kernel/hostname  
   
$ hostname  
new-machinename.domainname.com  
   
这样，通过修改/proc 文件系统中的文件，我们可以修改主机名。很多其他可配置的文件存在于/proc/sys/kernel/。这里不可能列出所有这些文件，读者可以自己去这个目录查看以得到更多细节信息。  
另一个可配置的目录是/proc/sys/net。这个目录中的文件可以用于修改机器/网络的网络属性。比如，简单修改一个文件，你可以在网络上瘾藏匿的计算机。  
   
$ echo 1 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_all  
   
这将在网络上瘾藏你的机器，因为它不响应icmp\_echo。主机将不会响应其他主机发出的ping 查询。  
   
$ ping machinename.domainname.com  
no answer from machinename.domainname.com  
   
要改回缺省设置，只要  
   
$ echo 0 > /proc/sys/net/ipv4/icmp\_echo\_ignore\_all  
   
/proc/sys 下还有许多其它可以用于改变内核属性。读者可以通过参考文献[1], [2] 获取更多信息。  
   
结论  
   
/proc 文件系统提供了一个基于文件的Linux 内部接口。它可以用于确定系统的各种不同设备和进程的状态。对他们进行配置。因而，理解和应用有关这个文件系统的知识是理解你的Linux 系统的关键。  
   
原文：http://www.sudu.cn/info/html/edu/20070101/292381.html  
   
什么是proc文件系统  
proc文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。用户和应用程序可以通过proc得到系统的信息，并可以改变内核的某些参数。由于系统的信息，如进程，是动态改变的，所以用户或应用程序读取proc文件时，proc文件系统是动态从系统内核读出所需信息并提交的。它的目录结构如下：  
目录名称 目录内容  
apm 高级电源管理信息  
cmdline 内核命令行  
Cpuinfo 关于Cpu信息  
Devices 可以用到的设备（块设备/字符设备）  
Dma 使用的DMA通道  
Filesystems 支持的文件系统  
Interrupts 中断的使用  
Ioports I/O端口的使用  
Kcore 内核核心印象  
Kmsg 内核消息  
Ksyms 内核符号表  
Loadavg 负载均衡  
Locks 内核锁  
Meminfo 内存信息  
Misc 杂项  
Modules 加载模块列表  
Mounts 加载的文件系统  
Partitions 系统识别的分区表  
Rtc 实时时钟  
Slabinfo Slab池信息  
Stat 全面统计状态表  
Swaps 对换空间的利用情况  
Version 内核版本  
Uptime 系统正常运行时间  
并不是所有这些目录在你的系统中都有，这取决于你的内核配置和装载的模块。另外，在/proc下还有三个很重要的目录：net，scsi和sys。Sys目录是可写的，可以通过它来访问或修改内核的参数（见下一部分），而net和scsi则依赖于内核配置。例如，如果系统不支持scsi，则scsi 目录不存在。  
除了以上介绍的这些，还有的是一些以数字命名的目录，它们是进程目录。系统中当前运行的每一个进程都有对应的一个目录在/proc下，以进程的PID号为目录名，它们是读取进程信息的接口。而self目录则是读取进程本身的信息接口，是一个link。Proc文件系统的名字就是由之而起。进程目录的结构如下：  
目录名称 目录内容  
Cmdline 命令行参数  
Environ 环境变量值  
Fd 一个包含所有文件描述符的目录  
Mem 进程的内存被利用情况  
Stat 进程状态  
Status 进程当前状态，以可读的方式显示出来  
Cwd 当前工作目录的链接  
Exe 指向该进程的执行命令文件  
Maps 内存映象  
Statm 进程内存状态信息  
Root 链接此进程的root目录  
用户如果要查看系统信息，可以用cat命令。例如：  
# cat /proc/interrupts  
CPU0  
0: 8728810 XT-PIC timer  
1: 8Array5 XT-PIC keyboard  
2: 0 XT-PIC cascade  
3: 5316Array5 XT-PIC aha152x  
4: 2014133 XT-PIC serial  
5: 44401 XT-PIC pcnet\_cs  
8: 2 XT-PIC rtc  
11: 8 XT-PIC i82365  
12: 182Array18 XT-PIC Mouse  
13: 1 XT-PIC fpu PS/2  
14: 1232265 XT-PIC ide0  
15: 7 XT-PIC ide1  
NMI: 0  
用户还可以实现修改内核参数。在/proc文件系统中有一个有趣的目录：/proc/sys。它不仅提供了内核信息，而且可以通过它修改内核参数，来优化你的系统。但是你必须很小心，因为可能会造成系统崩溃。最好是先找一台无关紧要的机子，调试成功后再应用到你的系统上。  
要改变内核的参数，只要用vi编辑或echo参数重定向到文件中即可。下面有一个例子：  
# cat /proc/sys/fs/file-max  
40Array6  
# echo 81Array2 > /proc/sys/fs/file-max  
# cat /proc/sys/fs/file-max  
81Array2  
如果你优化了参数，则可以把它们写成添加到文件rc.local中，使它在系统启动时自动完成修改。  
/proc文件系统中网络参数  
/proc/sys/kernel  
内核共享内存限制,默认最小为4M.两个参数可以改成一样.加大.  
/proc/sys/kernel/shmall 40Array6  
/proc/sys/kernel/shmmax 33554432  
   
/proc/sys/net/ipv4/  
该目录包含的是和tcp/ip协议相关的各种参数，下面我们就对这些网络参数加以详细的说明。  
ip\_forward 参数类型：BOOLEAN  
0 - 关闭(默认值)  
not 0 - 打开ip转发  
在网络本地接口之间转发数据报。该参数非常特殊，对该参数的修改将导致其它所有相关配置参数恢复其默认值(对于主机参阅RFC1122，对于路由器参见RFC1812)  
/proc/sys/net/ipv4/ip\_conntrack\_max 65440 (默认)  
ip\_default\_ttl 参数类型：INTEGER  
默认值为64 。表示IP数据报的Time To Live值。  
ip\_no\_pmtu\_disc 参数类型：BOOLEAN  
关闭路径MTU探测，默认值为FALSE  
ipfrag\_high\_thresh 参数类型：整型  
用来组装分段的IP包的最大内存量。当ipfrag\_high\_thresh数量的内存被分配来用来组装IP包，则IP分片处理器将丢弃数据报直到ipfrag\_low\_thresh数量的内存被用来组装IP包。  
ipfrag\_low\_thresh 参数类型：整型  
参见ipfrag\_high\_thresh。  
ipfrag\_time 参数类型：整型  
保存一个IP分片在内存中的时间。  
inet\_peer\_threshold 参数类型：整型  
INET对端存储器某个合适值，当超过该阀值条目将被丢弃。该阀值同样决定生存时间以及废物收集通过的时间间隔。条目越多?存活期越低?GC 间隔越短  
inet\_peer\_minttl 参数类型：整型  
条目的最低存活期。在重组端必须要有足够的碎片(fragment)存活期。这个最低存活期必须保证缓冲池容积是否少于inet\_peer\_threshold。该值以jiffies为单位测量。  
inet\_peer\_maxttl 参数类型：整型  
条目的最大存活期。在此期限到达之后?如果缓冲池没有耗尽压力的话(例如?缓冲池中的条目数目非常少)?不使用的条目将会超时。该值以jiffies为单位测量。  
inet\_peer\_gc\_mintime 参数类型：整型  
废物收集(GC)通过的最短间隔。这个间隔会影响到缓冲池中内存的高压力。 该值以jiffies为单位测量。  
inet\_peer\_gc\_maxtime 参数类型：整型  
废物收集(GC)通过的最大间隔，这个间隔会影响到缓冲池中内存的低压力。 该值以jiffies为单位测量。  
tcp\_syn\_retries 参数类型：整型  
对于一个新建连接，内核要发送多少个SYN 连接请求才决定放弃。不应该大于255，默认值是5，对应于180秒左右。  
tcp\_synack\_retries 参数类型：整型  
对于远端的连接请求SYN，内核会发送SYN ＋ACK数据报，以确认收到上一个SYN连接请求包。这是所谓的三次握手( threeway handshake)机制的第二个步骤。这里决定内核在放弃连接之前所送出的SYN+ACK 数目。  
tcp\_keepalive\_time 参数类型：整型  
当keepalive打开的情况下，TCP发送keepalive消息的频率，默认值是2个小时。  
tcp\_keepalive\_probes 参数类型：整型  
TCP发送keepalive探测以确定该连接已经断开的次数，默认值是Array。  
tcp\_keepalive\_interval 参数类型：整型  
探测消息发送的频率，乘以tcp\_keepalive\_probes就得到对于从开始探测以来没有响应的连接杀除的时间。默认值为75秒，也就是没有活动的连接将在大约11分钟以后将被丢弃。  
tcp\_retries1 参数类型：整型  
当出现可疑情况而必须向网络层报告这个可疑状况之前?需要进行多少次重试。最低的RFC 数值是3 ?这也是默认值?根据RTO的值大约在3秒- 8分钟之间。  
tcp\_retries2 参数类型：整型  
在丢弃激活的TCP连接之前?需要进行多少次重试。RFC1122规定，该值必须大于100秒。默认值为15，根据RTO的值来决定，相当于13-30分钟，  
tcp\_orphan\_retries 参数类型：整型  
在近端丢弃TCP连接之前?要进行多少次重试。默认值是7 个?相当于50秒- 16分钟?视RTO 而定。如果您的系统是负载很大的web服务器?那么也许需要降低该值?这类sockets 可能会耗费大量的资源。另外参的考tcp\_max\_orphans 。  
tcp\_fin\_timeout 参数类型：整型  
对于本端断开的socket连接，TCP保持在FIN-WAIT-2状态的时间。对方可能会断开连接或一直不结束连接或不可预料的进程死亡。默认值为60 秒。过去在2.2版本的内核中是180 秒。您可以设置该值?但需要注意?如果您的机器为负载很重的web服务器?您可能要冒内存被大量无效数据报填满的风险?FIN-WAIT-2 sockets 的危险性低于FIN-WAIT-1 ?因为它们最多只吃1.5K 的内存?但是它们存在时间更长。另外参考tcp\_max\_orphans。  
tcp\_max\_tw\_buckets 参数类型：整型  
系统在同时所处理的最大timewait sockets 数目。如果超过此数的话?time-wait socket 会被立即砍除并且显示警告信息。之所以要设定这个限制?纯粹为了抵御那些简单的DoS 攻击?千万不要人为的降低这个限制?不过?如果网络条件需要比默认值更多?则可以提高它(或许还要增加内存)。  
tcp\_tw\_recycle 参数类型：布尔  
打开快速TIME-WAIT sockets 回收。默认值是1。除非得到技术专家的建议或要求?请不要随意修改这个值。  
tcp\_max\_orphans 参数类型：整型  
系统所能处理不属于任何进程的TCP sockets最大数量。假如超过这个数量?那么不属于任何进程的连接会被立即reset，并同时显示警告信息。之所以要设定这个限制?纯粹为了抵御那些简单的DoS 攻击?千万不要依赖这个或是人为的降低这个限制  
tcp\_abort\_on\_overflow 参数类型：布尔  
当守护进程太忙而不能接受新的连接，就象对方发送reset消息，默认值是false。这意味着当溢出的原因是因为一个偶然的猝发，那么连接将恢复状态。只有在你确信守护进程真的不能完成连接请求时才打开该选项，该选项会影响客户的使用。  
tcp\_syncookies 参数类型：整型  
只有在内核编译时选择了CONFIG\_SYNCOOKIES时才会发生作用。当出现syn等候队列出现溢出时象对方发送syncookies。目的是为了防止syn flood攻击。默认值是false。  
注意：该选项千万不能用于那些没有收到攻击的高负载服务器，如果在日志中出现synflood消息，但是调查发现没有收到synflood攻击，而是合法用户的连接负载过高的原因，你应该调整其它参数来提高服务器性能。参考: tcp\_max\_syn\_backlog, tcp\_synack\_retries, tcp\_abort\_on\_overflow.  
syncookie严重的违背TCP协议，不允许使用TCP扩展，可能对某些服务导致严重的性能影响(如SMTP转发)。  
tcp\_stdurg 参数类型：整型  
使用TCP urg pointer 字段中的主机请求解释功能。大部份的主机都使用老旧的BSD解释，因此如果您在Linux 打开它?或会导致不能和它们正确沟通。默认值为为?FALSE  
tcp\_max\_syn\_backlog 参数类型：整型  
对于那些依然还未获得客户端确认的连接请求?需要保存在队列中最大数目。对于超过128Mb 内存的系统?默认值是1024 ?低于128Mb 的则为128。如果服务器经常出现过载?可以尝试增加这个数字。警告?假如您将此值设为大于1024?最好修改include/net/tcp.h 里面的TCP\_SYNQ\_HSIZE ?以保持TCP\_SYNQ\_HSIZE\*16 0)或者bytes-bytes/2^(-tcp\_adv\_win\_scale)(如果tcp\_adv\_win\_scale  128Mb 32768-61000  
0)则系统将忽略所有发送给自己的ICMP ECHO请求或那些广播地址的请求。  
icmp\_destunreach\_rate - 整数  
icmp\_paramprob\_rate - 整数  
icmp\_timeexceed\_rate - 整数  
icmp\_echoreply\_rate - 整数(not enabled per default)  
限制发向特定目标的ICMP数据报的最大速率。0表示没有任何限制，否则表示jiffies数据单位中允许发送的个数。  
icmp\_ignore\_bogus\_error\_responses - 布尔类型  
某些路由器违背RFC1122标准，其对广播帧发送伪造的响应来应答。这种违背行为通常会被以告警的方式记录在系统日志中。如果该选项设置为True，内核不会记录这种警告信息。默认值为False。  
(1) Jiffie: 内核使用的内部时间单位，在i386系统上大小为1/100s，在Alpha中为1/1024S。在/usr/include/asm/param.h中的HZ定义有特定系统的值。  
conf/interface/\*:  
conf/all/\*是特定的，用来修改所有接口的设置，is special and changes the settings for all interfaces.  
Change special settings per interface.  
log\_martians - 布尔类型  
记录带有不允许的地址的数据报到内核日志中。  
accept\_redirects - 布尔类型  
收发接收ICMP重定向消息。对于主机来说默认为True，对于用作路由器时默认值为False。  
forwarding - 布尔类型  
在该接口打开转发功能  
mc\_forwarding - 布尔类型  
是否进行多播路由。只有内核编译有CONFIG\_MROUTE并且有路由服务程序在运行该参数才有效。  
proxy\_arp - 布尔类型  
打开proxy arp功能。  
shared\_media - 布尔类型  
发送(路由器)或接收(主机) RFC1620 共享媒体重定向。覆盖ip\_secure\_redirects的值。默认为True。  
secure\_redirects - 布尔类型  
仅仅接收发给默认网关列表中网关的ICMP重定向消息，默认值是TRUE。  
send\_redirects - 布尔类型  
如果是router，发送重定向消息，默认值是TRUE  
bootp\_relay - 布尔类型  
接收源地址为0.b.c.d，目的地址不是本机的数据报。用来支持BOOTP转发服务进程，该进程将捕获并转发该包。默认为False，目前还没有实现。  
accept\_source\_route - 布尔类型  
接收带有SRR选项的数据报。对于主机来说默认为False，对于用作路由器时默认值为True。  
rp\_filter 参数类型  
1 - 通过反向路径回溯进行源地址验证(在RFC1812中定义)。对于单穴主机和stub网络路由器推荐使用该选项。  
0 - 不通过反向路径回溯进行源地址验证。  
默认值为0。某些发布在启动时自动将其打开。  
/proc/sys/net/ipv4/netfilter  
ip\_conntrack\_buckets  
ip\_conntrack\_count  
ip\_conntrack\_generic\_timeout  
ip\_conntrack\_icmp\_timeout  
ip\_conntrack\_log\_invalid  
ip\_conntrack\_max 65536(默认)  
ip\_conntrack\_tcp\_be\_liberal  
ip\_conntrack\_tcp\_loose  
ip\_conntrack\_tcp\_max\_retrans  
ip\_conntrack\_tcp\_timeout\_close  
ip\_conntrack\_tcp\_timeout\_close\_wait  
ip\_conntrack\_tcp\_timeout\_established 432000(单位为秒,5天)  
ip\_conntrack\_tcp\_timeout\_fin\_wait  
ip\_conntrack\_tcp\_timeout\_last\_ack  
ip\_conntrack\_tcp\_timeout\_max\_retrans  
ip\_conntrack\_tcp\_timeout\_syn\_recv  
ip\_conntrack\_tcp\_timeout\_syn\_sent  
ip\_conntrack\_tcp\_timeout\_time\_wait  
ip\_conntrack\_udp\_timeout  
ip\_conntrack\_udp\_timeout\_stream  
   
#  
   
#  
#  
   
   
Linux Linux Linux Linux 中/proc /proc /proc /proc 目录下文件详解  
声明：可以自由转载本文, 但请务必保留本文的完整性。  
作者：张子坚  
email:zhangzijian@163.com  
说明：本文所涉及示例均在fedora core3 下得到。  
--------------------------------------------------------------------------------  
/proc 文件系统下的多种文件提供的系统信息不是针对某个特定进程的, 而是能够在整个系统范围的上下文  
中使用。可以使用的文件随系统配置的变化而变化。命令procinfo 能够显示基于其中某些文件的多种系统  
信息。  
以下详细描述/proc 下的文件:  
--------------------------------------------------------------------------------  
/proc/cmdline 文件  
这个文件给出了内核启动的命令行。它和用于进程的cmdline 项非常相似。  
示例：  
[root@localhost proc]# cat cmdline  
ro root=LABEL=/ rhgb quiet  
--------------------------------------------------------------------------------  
/proc/cpuinfo 文件  
这个文件提供了有关系统CPU 的多种信息。这些信息是从内核里对CPU 的测试代码中得到的。文件列出  
了CPU 的普通型号 （386,486,586,686 等 ）, 以及能得到的更多特定信息 （ 制造商, 型号和版本 ） 。 文件还包含  
了以bogomips 表示的处理器速度, 而且如果检测到CPU 的多种特性或者bug, 文件还会包含相应的标志。这  
个文件的格式为：文件由多行构成, 每行包括一个域名称, 一个冒号和一个值。  
示例：  
[root@localhost proc]# cat cpuinfo  
processor : 0  
vendor\_id : AuthenticAMD  
cpu family : 6  
model : 8  
model name : AMD Athlon(tm) XP 1800+  
stepping : 1  
cpu MHz : 1530.165  
cache size : 256 KB  
fdiv\_bug : no  
hlt\_bug : no  
f00f\_bug : nocoma\_bug : no  
fpu : yes  
fpu\_exception : yes  
cpuid level : 1  
wp : yes  
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36 mmx fxsr sse syscall mmxext  
3dnowext 3dnow  
bogomips : 2998.27  
--------------------------------------------------------------------------------  
/proc/devices 文件  
这个文件列出字符和块设备的主设备号, 以及分配到这些设备号的设备名称。  
示例：  
[root@localhost /]# cat /proc/devices  
Character devices:  
1 mem  
4 /dev/vc/0  
4 tty  
4 ttyS  
5 /dev/tty  
5 /dev/console  
5 /dev/ptmx  
6 lp  
7 vcs  
10 misc  
13 input  
14 sound  
29 fb  
36 netlink  
116 alsa  
128 ptm  
136 pts  
180 usb  
Block devices:  
1 ramdisk  
2 fd  
3 ide0  
9 md  
22 ide1  
253 device-mapper  
254 mdp  
--------------------------------------------------------------------------------/proc/dma 文件  
这个文件列出由驱动程序保留的DMA 通道和保留它们的驱动程序名称 。casade 项供用于把次DMA 控制器  
从主控制器分出的DMA 行所使用; 这一行不能用于其它用途。  
示例：  
[root@localhost ~]# cat /proc/dma  
4: cascade  
--------------------------------------------------------------------------------  
/proc/filesystems 文件  
这个文件列出可供使用的文件系统类型, 一种类型一行 。 虽然它们通常是编入内核的文件系统类型, 但该文件  
还可以包含可加载的内核模块加入的其它文件系统类型。  
示例：  
[root@localhost proc]# cat /proc/filesystems  
nodev sysfs  
nodev rootfs  
nodev bdev  
nodev proc  
nodev sockfs  
nodev binfmt\_misc  
nodev usbfs  
nodev usbdevfs  
nodev futexfs  
nodev tmpfs  
nodev pipefs  
nodev eventpollfs  
nodev devpts  
ext2  
nodev ramfs  
nodev hugetlbfs  
iso9660  
nodev mqueue  
nodev selinuxfs  
ext3  
nodev rpc\_pipefs  
nodev autofs  
--------------------------------------------------------------------------------  
/proc/interrupts 文件  
这个文件的每一行都有一个保留的中断 。 每行中的域有 ： 中断号, 本行中断的发生次数, 可能带有一个加号的  
域（SA\_INTERRUPT 标志设置）, 以及登记这个中断的驱动程序的名字。可以在安装新硬件前, 像查看  
/proc/dma 和/proc/ioports 一样用cat 命令手工查看手头的这个文件 。 这几个文件列出了当前投入使用的资源  
（但是不包括那些没有加载驱动程序的硬件所使用的资源 ） 。  
示例：  
[root@localhost SPECS]# cat /proc/interruptsCPU0  
0: 7039406 XT-PIC timer  
1: 6533 XT-PIC i8042  
2: 0 XT-PIC cascade  
3: 0 XT-PIC uhci\_hcd  
5: 108 XT-PIC VIA8233, uhci\_hcd  
8: 1 XT-PIC rtc  
9: 0 XT-PIC acpi  
10: 0 XT-PIC ehci\_hcd  
11: 17412 XT-PIC uhci\_hcd, eth0  
12: 140314 XT-PIC i8042  
14: 37897 XT-PIC ide0  
15: 60813 XT-PIC ide1  
NMI: 0  
ERR: 1  
--------------------------------------------------------------------------------  
/proc/ioports 文件  
这个文件列出了诸如磁盘驱动器, 以太网卡和声卡设备等多种设备驱动程序登记的许多I/O 端口范围。  
示例：  
[root@localhost SPECS]# cat /proc/ioports  
0000-001f : dma1  
0020-0021 : pic1  
0040-0043 : timer0  
0050-0053 : timer1  
0060-006f : keyboard  
0070-0077 : rtc  
0080-008f : dma page reg  
00a0-00a1 : pic2  
00c0-00df : dma2  
00f0-00ff : fpu  
0170-0177 : ide1  
01f0-01f7 : ide0  
0376-0376 : ide1  
0378-037a : parport0  
037b-037f : parport0  
03c0-03df : vga+  
03f6-03f6 : ide0  
03f8-03ff : serial  
0800-0803 : PM1a\_EVT\_BLK  
0804-0805 : PM1a\_CNT\_BLK  
0808-080b : PM\_TMR  
0810-0815 : ACPI CPU throttle  
0820-0823 : GPE0\_BLK0cf8-0cff : PCI conf1  
dc00-dcff : 0000:00:12.0  
dc00-dcff : via-rhine  
e000-e0ff : 0000:00:11.5  
e000-e0ff : VIA8233  
e400-e41f : 0000:00:10.0  
e400-e41f : uhci\_hcd  
e800-e81f : 0000:00:10.1  
e800-e81f : uhci\_hcd  
ec00-ec1f : 0000:00:10.2  
ec00-ec1f : uhci\_hcd  
fc00-fc0f : 0000:00:11.1  
fc00-fc07 : ide0  
fc08-fc0f : ide1  
--------------------------------------------------------------------------------  
/proc/kcore 文件  
这个文件是系统的物理内存以core 文件格式保存的文件。例如,GDB 能用它考察内核的数据结构。它不是  
纯文本, 而是/proc 目录下为数不多的几个二进制格式的项之一。  
示例：  
暂无  
--------------------------------------------------------------------------------  
/proc/kmsg 文件  
这个文件用于检索用printk 生成的内核消息。任何时刻只能有一个具有超级用户权限的进程可以读取这个  
文件。也可以用系统调用syslog 检索这些消息。通常使用工具dmesg 或守护进程klogd 检索这些消息。  
示例：  
暂无  
--------------------------------------------------------------------------------  
/proc/ksyms 文件  
这个文件列出了已经登记的内核符号; 这些符号给出了变量或函数的地址 。 每行给出一个符号的地址, 符号名  
称以及登记这个符号的模块。程序ksyms,insmod 和kmod 使用这个文件。它还列出了正在运行的任务数,  
总任务数和最后分配的PID 。  
示例：  
暂无  
--------------------------------------------------------------------------------  
/proc/loadavg 文件  
这个文件给出以几个不同的时间间隔计算的系统平均负载, 这就如同uptime 命令显示的结果那样 。 前三个数  
字是平均负载 。 这是通过计算过去1 分钟,5 分钟,15 分钟里运行队列中的平均任务数得到的 。 随后是正在运行  
的任务数和总任务数。最后是上次使用的进程号。  
示例：[root@localhost ~]# cat /proc/loadavg  
0.11 0.16 0.14 3/126 3912  
--------------------------------------------------------------------------------  
/proc/locks 文件  
这个文件包含在打开的文件上的加锁信息。文件中的每一行描述了特定文件和文档上的加锁信息以及对文  
件施加的锁的类型。内核也可以需要时对文件施加强制性锁。  
示例：  
[root@localhost redhat]# cat /proc/locks  
1: POSIX ADVISORY READ 3822 03:0a:1067117 0 EOF  
2: POSIX ADVISORY READ 3822 03:0a:1067138 0 EOF  
3: POSIX ADVISORY WRITE 3326 03:0a:2326540 0 EOF  
4: POSIX ADVISORY WRITE 2639 03:0a:2966595 0 EOF  
5: FLOCK ADVISORY WRITE 2591 03:0a:2966586 0 EOF  
6: POSIX ADVISORY WRITE 2540 03:0a:2966578 0 EOF  
7: POSIX ADVISORY WRITE 2530 03:0a:2966579 0 EOF  
8: POSIX ADVISORY WRITE 2402 03:0a:2966563 0 EOF  
9: POSIX ADVISORY WRITE 2371 03:0a:2966561 0 EOF  
--------------------------------------------------------------------------------  
/proc/mdstat 文件  
这个文件包含了由md 设备驱动程序控制的RAID 设备信息。  
示例：  
[root@localhost ~]# cat /proc/mdstat  
Personalities :  
unused devices: <none>  
--------------------------------------------------------------------------------  
/proc/meminfo 文件  
这个文件给出了内存状态的信息 。 它显示出系统中空闲内存, 已用物理内存和交换内存的总量 。 它还显示出  
内核使用的共享内存和缓冲区总量。这些信息的格式和free 命令显示的结果类似。  
示例：  
[root@localhost ~]# cat /proc/meminfo  
MemTotal: 223812 kB  
MemFree: 3764 kB  
Buffers: 9148 kB  
Cached: 92112 kB  
SwapCached: 364 kB  
Active: 183640 kB  
Inactive: 17196 kB  
HighTotal: 0 kB  
HighFree: 0 kB  
LowTotal: 223812 kB  
LowFree: 3764 kBSwapTotal: 626524 kB  
SwapFree: 620328 kB  
Dirty: 12 kB  
Writeback: 0 kB  
Mapped: 142880 kB  
Slab: 12668 kB  
Committed\_AS: 376732 kB  
PageTables: 2336 kB  
VmallocTotal: 3907576 kB  
VmallocUsed: 2968 kB  
VmallocChunk: 3904224 kB  
HugePages\_Total: 0  
HugePages\_Free: 0  
Hugepagesize: 4096 kB  
--------------------------------------------------------------------------------  
/proc/misc 文件  
这个文件报告用内核函数misc\_register 登记的设备驱动程序。  
示例：  
[root@localhost ~]# cat /proc/misc  
63 device-mapper  
175 agpgart  
135 rtc  
--------------------------------------------------------------------------------  
/proc/modules 文件  
这个文件给出可加载内核模块的信息 。lsmod 程序用这些信息显示有关模块的名称, 大小, 使用数目方面的信  
息。  
示例：  
[root@localhost /]# cat /proc/modules  
md5 4033 1 - Live 0x10a7f000  
ipv6 232577 8 - Live 0x10b0c000  
parport\_pc 24705 1 - Live 0x10a8b000  
lp 11565 0 - Live 0x10a7b000  
parport 41737 2 parport\_pc,lp, Live 0x10a55000  
autofs4 24005 0 - Live 0x10a74000  
i2c\_dev 10433 0 - Live 0x109d2000  
i2c\_core 22081 1 i2c\_dev, Live 0x10a6d000  
sunrpc 160421 1 - Live 0x10a9d000  
ipt\_REJECT 6465 1 - Live 0x109da000  
ipt\_state 1857 5 - Live 0x109eb000  
ip\_conntrack 40693 1 ipt\_state, Live 0x10a62000  
iptable\_filter 2753 1 - Live 0x10896000  
ip\_tables 16193 3 ipt\_REJECT,ipt\_state,iptable\_filter, Live 0x109ed000dm\_mod 54741 0 - Live 0x109f8000  
button 6481 0 - Live 0x10905000  
battery 8517 0 - Live 0x109d6000  
ac 4805 0 - Live 0x10908000  
uhci\_hcd 31449 0 - Live 0x109dd000  
ehci\_hcd 31557 0 - Live 0x10949000  
snd\_via82xx 27237 2 - Live 0x10953000  
snd\_ac97\_codec 64401 1 snd\_via82xx, Live 0x10912000  
snd\_pcm\_oss 47609 0 - Live 0x1093c000  
snd\_mixer\_oss 17217 2 snd\_pcm\_oss, Live 0x1090c000  
snd\_pcm 97993 2 snd\_via82xx,snd\_pcm\_oss, Live 0x10923000  
snd\_timer 29765 1 snd\_pcm, Live 0x108ec000  
snd\_page\_alloc 9673 2 snd\_via82xx,snd\_pcm, Live 0x108bd000  
gameport 4801 1 snd\_via82xx, Live 0x108a6000  
snd\_mpu401\_uart 8769 1 snd\_via82xx, Live 0x108b9000  
snd\_rawmidi 26725 1 snd\_mpu401\_uart, Live 0x108e4000  
snd\_seq\_device 8137 1 snd\_rawmidi, Live 0x1083b000  
snd 54053 11  
snd\_via82xx,snd\_ac97\_codec,snd\_pcm\_oss,snd\_mixer\_oss,snd\_pcm,snd\_timer,snd\_mpu401\_uart,snd\_rawmidi,sn  
d\_seq\_device, Live 0x108f6000  
soundcore 9889 2 snd, Live 0x1089b000  
via\_rhine 23497 0 - Live 0x1089f000  
mii 4673 1 via\_rhine, Live 0x10893000  
floppy 58609 0 - Live 0x108a9000  
ext3 116809 1 - Live 0x10875000  
jbd 74969 1 ext3, Live 0x10861000  
lsmod 命令显示结果如下：  
[root@localhost /]# lsmod  
Module Size Used by  
md5 4033 1  
ipv6 232577 8  
parport\_pc 24705 1  
lp 11565 0  
parport 41737 2 parport\_pc,lp  
autofs4 24005 0  
i2c\_dev 10433 0  
i2c\_core 22081 1 i2c\_dev  
sunrpc 160421 1  
ipt\_REJECT 6465 1  
ipt\_state 1857 5  
ip\_conntrack 40693 1 ipt\_state  
iptable\_filter 2753 1ip\_tables 16193 3 ipt\_REJECT,ipt\_state,iptable\_filter  
dm\_mod 54741 0  
button 6481 0  
battery 8517 0  
ac 4805 0  
uhci\_hcd 31449 0  
ehci\_hcd 31557 0  
snd\_via82xx 27237 2  
snd\_ac97\_codec 64401 1 snd\_via82xx  
snd\_pcm\_oss 47609 0  
snd\_mixer\_oss 17217 2 snd\_pcm\_oss  
snd\_pcm 97993 2 snd\_via82xx,snd\_pcm\_oss  
snd\_timer 29765 1 snd\_pcm  
snd\_page\_alloc 9673 2 snd\_via82xx,snd\_pcm  
gameport 4801 1 snd\_via82xx  
snd\_mpu401\_uart 8769 1 snd\_via82xx  
snd\_rawmidi 26725 1 snd\_mpu401\_uart  
snd\_seq\_device 8137 1 snd\_rawmidi  
snd 54053 11  
snd\_via82xx,snd\_ac97\_codec,snd\_pcm\_oss,snd\_mixer\_oss,snd\_pcm,snd\_timer,snd\_mpu401\_uart,snd\_rawmidi,sn  
d\_seq\_device  
soundcore 9889 2 snd  
via\_rhine 23497 0  
mii 4673 1 via\_rhine  
floppy 58609 0  
ext3 116809 1  
jbd 74969 1 ext3  
--------------------------------------------------------------------------------  
/proc/mounts 文件  
这个文件以/etc/mtab 文件的格式给出当前系统所安装的文件系统信息。这个文件也能反映出任何手工安装  
从而在/etc/mtab 文件中没有包含的文件系统。  
示例：  
[root@localhost /]# cat /proc/mounts  
rootfs / rootfs rw 0 0  
/proc /proc proc rw,nodiratime 0 0  
none /dev tmpfs rw 0 0  
/dev/root / ext3 rw 0 0  
none /dev tmpfs rw 0 0  
none /selinux selinuxfs rw 0 0  
/proc /proc proc rw,nodiratime 0 0  
/proc/bus/usb /proc/bus/usb usbfs rw 0 0  
/sys /sys sysfs rw 0 0none /dev/pts devpts rw 0 0  
none /dev/shm tmpfs rw 0 0  
none /proc/sys/fs/binfmt\_misc binfmt\_misc rw 0 0  
sunrpc /var/lib/nfs/rpc\_pipefs rpc\_pipefs rw 0 0  
--------------------------------------------------------------------------------  
/proc/pci 文件  
这个文件给出PCI 设备的信息。用它可以方便地诊断PCI 问题。你可以从这个文件中检索到的信息包括诸  
如IDE 接口或USB 控制器这样的设备, 总线, 设备和功能编号, 设备延迟以及IRQ 编号。  
示例：  
[root@localhost /]# cat /proc/pci  
PCI devices found:  
Bus 0, device 0, function 0:  
Class 0600: PCI device 1106:3116 (rev 0).  
Master Capable. Latency=8.  
Prefetchable 32 bit memory at 0xe0000000 [0xe7ffffff].  
Bus 0, device 1, function 0:  
Class 0604: PCI device 1106:b091 (rev 0).  
Master Capable. No bursts. Min Gnt=12.  
Bus 0, device 16, function 2:  
Class 0c03: PCI device 1106:3038 (rev 12.  
IRQ 5.  
Master Capable. Latency=32.  
I/O at 0xec00 [0xec1f].  
Bus 0, device 16, function 1:  
Class 0c03: PCI device 1106:3038 (rev 12.  
IRQ 3.  
Master Capable. Latency=32.  
I/O at 0xe800 [0xe81f].  
Bus 0, device 16, function 0:  
Class 0c03: PCI device 1106:3038 (rev 12.  
IRQ 11.  
Master Capable. Latency=32.  
I/O at 0xe400 [0xe41f].  
Bus 0, device 16, function 3:  
Class 0c03: PCI device 1106:3104 (rev 130).  
IRQ 10.  
Master Capable. Latency=32.  
Non-prefetchable 32 bit memory at 0xdfffff00 [0xdfffffff].  
Bus 0, device 17, function 0:  
Class 0601: PCI device 1106:3177 (rev 0).  
Bus 0, device 17, function 1:Class 0101: PCI device 1106:0571 (rev 6).  
IRQ 255.  
Master Capable. Latency=32.  
I/O at 0xfc00 [0xfc0f].  
Bus 0, device 17, function 5:  
Class 0401: PCI device 1106:3059 (rev 80).  
IRQ 5.  
I/O at 0xe000 [0xe0ff].  
Bus 0, device 18, function 0:  
Class 0200: PCI device 1106:3065 (rev 116).  
IRQ 11.  
Master Capable. Latency=32. Min Gnt=3.Max Lat=8.  
I/O at 0xdc00 [0xdcff].  
Non-prefetchable 32 bit memory at 0xdffffe00 [0xdffffeff].  
Bus 1, device 0, function 0:  
Class 0300: PCI device 5333:8d04 (rev 0).  
IRQ 11.  
Master Capable. Latency=32. Min Gnt=4.Max Lat=255.  
Non-prefetchable 32 bit memory at 0xdfe80000 [0xdfefffff].  
Prefetchable 32 bit memory at 0xd0000000 [0xd7ffffff].  
--------------------------------------------------------------------------------  
/proc/stat 文件  
这个文件包含的信息有CPU 利用率, 磁盘, 内存页, 内存对换, 全部中断, 接触开关以及赏赐自举时间（自1970  
年1 月1 日起的秒数 ） 。  
示例：  
[root@localhost /]# cat /proc/stat  
cpu 31994 3898 7161 381600 15254 451 0  
cpu0 31994 3898 7161 381600 15254 451 0  
intr 4615930 4404290 3364 0 0 12 0 7 0 2 0 0 12618 112114 0 44142 39381  
ctxt 1310498  
btime 1148891913  
processes 4249  
procs\_running 4  
procs\_blocked 0  
--------------------------------------------------------------------------------  
/proc/uptime 文件  
这个文件给出自从上次系统自举以来的秒数, 以及其中有多少秒处于空闲 。 这主要供uptime 程序使用 。 比较  
这两个数字能够告诉你长期来看CPU 周期浪费的比例。  
示例：  
[root@localhost /]# cat /proc/uptime  
4477.04 4021.10--------------------------------------------------------------------------------  
/proc/version 文件  
这个文件只有一行内容, 说明正在运行的内核版本。可以用标准的[编程](http://www.2cto.com/kf)方法进行分析获得所需的系统信息 。  
示例：  
[root@localhost /]# cat /proc/version  
[Linux](http://www.2cto.com/os/linux/) version 2.6.9-1.667 ( bhcompile@tweety.build.redhat.com ) (gcc version 3.4.2 20041017 (Red Hat  
3.4.2-6.fc3)) #1 Tue Nov 2 14:41:25 EST 2004  
--------------------------------------------------------------------------------  
/proc/net 子目录  
此目录下的文件描述或修改了联网代码的行为。可以通过使用 arp,netstat,route 和ipfwadm 命令设置或查询  
这些特殊文件中的许多文件。  
示例：  
[root@localhost /]# ls /proc/net  
anycast6 ip\_conntrack mcfilter6 rt6\_stats tcp  
arp ip\_conntrack\_expect netlink rt\_acct tcp6  
dev ip\_mr\_cache netstat rt\_cache udp  
dev\_mcast ip\_mr\_vif packet snmp udp6  
dev\_snmp6 ip\_tables\_matches psched snmp6 unix  
if\_inet6 ip\_tables\_names raw sockstat wireless  
igmp ip\_tables\_targets raw6 sockstat6  
igmp6 ipv6\_route route softnet\_stat  
ip6\_flowlabel mcfilter rpc stat  
--------------------------------------------------------------------------------  
以下摘要介绍此目录下文件的功能：  
arp  
转储每个网络接口的arp 表中dev 包的统计  
dev  
来自网络设备的统计  
dev\_mcast  
列出二层（数据链路层）多播组  
igmp  
加入的IGMP 多播组  
netlink  
netlink 套接口的信息  
netstat  
网络流量的多种统计。第一行是信息头, 带有每个变量的名称。接下来的一行保存相应变量的值  
raw  
原始套接口的套接口表  
route  
静态路由表  
rpc  
包含RPC 信息的目录rt\_cache  
路由缓冲  
snmp  
snmp agent 的ip/icmp/tcp/udp 协议统计; 各行交替给出字段名和值  
sockstat  
列出使用的tcp/udp/raw/pac/syc\_cookies 的数量  
tcp  
TCP 连接的套接口  
udp  
UDP 连接的套接口表  
unix  
UNIX 域套接口的套接口表  
--------------------------------------------------------------------------------  
示例：[root@localhost /]# cat /proc/net/route  
Iface Destination Gateway Flags RefCnt Use Metric Mask MTU  
Window IRTT  
eth0 0035C2DA 00000000 0001 0 0 0 80FFFFF0  
eth0 0000FEA9 00000000 0001 0 0 0 0000FFF0  
eth0 00000000 0135C2DA 0003 0 0 0 00000000  
--------------------------------------------------------------------------------  
[root@localhost /]# cat /proc/net/tcp  
sl local\_address rem\_address st tx\_queue rx\_queue tr tm->when retrnsmt uid timeout inode  
0: 00000000:8000 00000000:0000 0A 00000000:00000000 00:00000000 00000000 29 0 9525 1  
0dde7500 3000 0 0 2 -1  
1: 00000000:006F 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 9484 1  
0dde79e0 3000 0 0 2 -1  
2: 0100007F:0277 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 10049 1  
0a8e3a00 3000 0 0 2 -1  
3: 0100007F:14D7 00000000:0000 0A 00000000:00000000 00:00000000 00000000 99 0 9847 1  
0dde7020 3000 0 0 2 -1  
4: 0100007F:0019 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 10286 1  
0a8e3520 3000 0 0 2 -1  
--------------------------------------------------------------------------------  
[root@localhost /]# cat /proc/net/arp  
IP address HW type Flags HW address Mask Device  
218.194.53.1 0x1 0x2 00:0D:BC:78:07:3F \* eth0  
--------------------------------------------------------------------------------  
[root@localhost /]# cat /proc/net/udp  
sl local\_address rem\_address st tx\_queue rx\_queue tr tm->when retrnsmt uid timeout inode  
0: 00000000:8000 00000000:0000 07 00000000:00000000 00:00000000 00000000 29 0 9520 2  
0b4ef7c0  
105: 00000000:14E9 00000000:0000 07 00000000:00000000 00:00000000 00000000 99 0 10284 20b4ef040  
111: 00000000:006F 00000000:0000 07 00000000:00000000 00:00000000 00000000 0 0 9483 2  
0b4efcc0  
116: 00000000:02F4 00000000:0000 07 00000000:00000000 00:00000000 00000000 0 0 9511 2  
0b4efa40  
119: 00000000:0277 00000000:0000 07 00000000:00000000 00:00000000 00000000 0 0 10050 2  
0b4ef2c0  
--------------------------------------------------------------------------------  
/proc/scsi 子目录  
此目录下包含一个列出了所有检测到的SCSI 设备的文件, 并且为每种控制器驱动程序提供一个目录, 在这个  
目录下又为已安装的此种控制器的每个实例提供一个子目录。  
示例：  
由于本人的机器没有SCSI 设备, 顾暂时无法提供示例。  
--------------------------------------------------------------------------------  
/proc/sys 子目录  
在此目录下有许多子目录 。 此目录中的许多项都可以用来调整系统的性能 。 这个目录包含信息太多, 无法介  
绍全部。只在示例中展示目录下的一些文件。  
示例：[root@localhost /]# ls /proc/sys  
debug dev fs kernel net proc sunrpc vm  
--------------------------------------------------------------------------------  
[root@localhost ~]# ls /proc/sys/fs  
aio-max-nr dentry-state file-nr lease-break-time overflowgid  
aio-nr dir-notify-enable inode-nr leases-enable overflowuid  
binfmt\_misc file-max inode-state mqueue quota  
--------------------------------------------------------------------------------  
[root@localhost ~]# ls /proc/sys/kernel  
acct hotplug panic sem  
cad\_pid modprobe panic\_on\_oops shmall  
cap-bound msgmax pid\_max shmmax  
core\_pattern msgmnb print-fatal-signals shmmni  
core\_uses\_pid msgmni printk sysrq  
ctrl-alt-del ngroups\_max printk\_ratelimit tainted  
domainname osrelease printk\_ratelimit\_burst threads-max  
exec-shield ostype pty vdso  
exec-shield-randomize overflowgid random version  
hostname overflowuid real-root-dev  
--------------------------------------------------------------------------------  
[root@localhost ~]# ls /proc/sys/net  
core ethernet ipv4 ipv6 unix  
--------------------------------------------------------------------------------  
[root@localhost sys]# ls /proc/sys/vmblock\_dump laptop\_mode nr\_pdflush\_threads  
dirty\_background\_ratio legacy\_va\_layout overcommit\_memory  
dirty\_expire\_centisecs lower\_zone\_protection overcommit\_ratio  
dirty\_ratio max\_map\_count page-cluster  
dirty\_writeback\_centisecs min\_free\_kbytes swappiness  
hugetlb\_shm\_group nr\_hugepages vfs\_cache\_pressure  
--------------------------------------------------------------------------------  
[root@localhost sys]# ls /proc/sys/net/ipv4  
conf tcp\_fack  
icmp\_echo\_ignore\_all tcp\_fin\_timeout  
icmp\_echo\_ignore\_broadcasts tcp\_frto  
icmp\_ignore\_bogus\_error\_responses tcp\_keepalive\_intvl  
icmp\_ratelimit tcp\_keepalive\_probes  
icmp\_ratemask tcp\_keepalive\_time  
igmp\_max\_memberships tcp\_low\_latency  
igmp\_max\_msf tcp\_max\_orphans  
inet\_peer\_gc\_maxtime tcp\_max\_syn\_backlog  
inet\_peer\_gc\_mintime tcp\_max\_tw\_buckets  
inet\_peer\_maxttl tcp\_mem  
inet\_peer\_minttl tcp\_moderate\_rcvbuf  
inet\_peer\_threshold tcp\_no\_metrics\_save  
ip\_autoconfig tcp\_orphan\_retries  
ip\_conntrack\_max tcp\_reordering  
ip\_default\_ttl tcp\_retrans\_collapse  
ip\_dynaddr tcp\_retries1  
ip\_forward tcp\_retries2  
ipfrag\_high\_thresh tcp\_rfc1337  
ipfrag\_low\_thresh tcp\_rmem  
ipfrag\_secret\_interval tcp\_sack  
ipfrag\_time tcp\_stdurg  
ip\_local\_port\_range tcp\_synack\_retries  
ip\_nonlocal\_bind tcp\_syncookies  
ip\_no\_pmtu\_disc tcp\_syn\_retries  
neigh tcp\_timestamps  
netfilter tcp\_tso\_win\_divisor  
route tcp\_tw\_recycle  
tcp\_abort\_on\_overflow tcp\_tw\_reuse  
tcp\_adv\_win\_scale tcp\_vegas\_alpha  
tcp\_app\_win tcp\_vegas\_beta  
tcp\_bic tcp\_vegas\_cong\_avoid  
tcp\_bic\_fast\_convergence tcp\_vegas\_gamma  
tcp\_bic\_low\_window tcp\_westwood  
tcp\_dsack tcp\_window\_scalingtcp\_ecn tcp\_wmem  
--------------------------------------------------------------------------------  
[root@localhost sys]# cat /proc/sys/kernel/shmall  
2097152  
--------------------------------------------------------------------------------  
[root@localhost sys]# cat /proc/sys/kernel/osrelease  
2.6.9-1.667  
--------------------------------------------------------------------------------  
总结 ：/proc 文件系统包含了大量的有关当前系统状态的信息 。proc 的手册页中也有对这些文件的解释文档 。  
把文件和分析这些文件的工具产生的输出进行比较能够更加清晰地了解这些文件